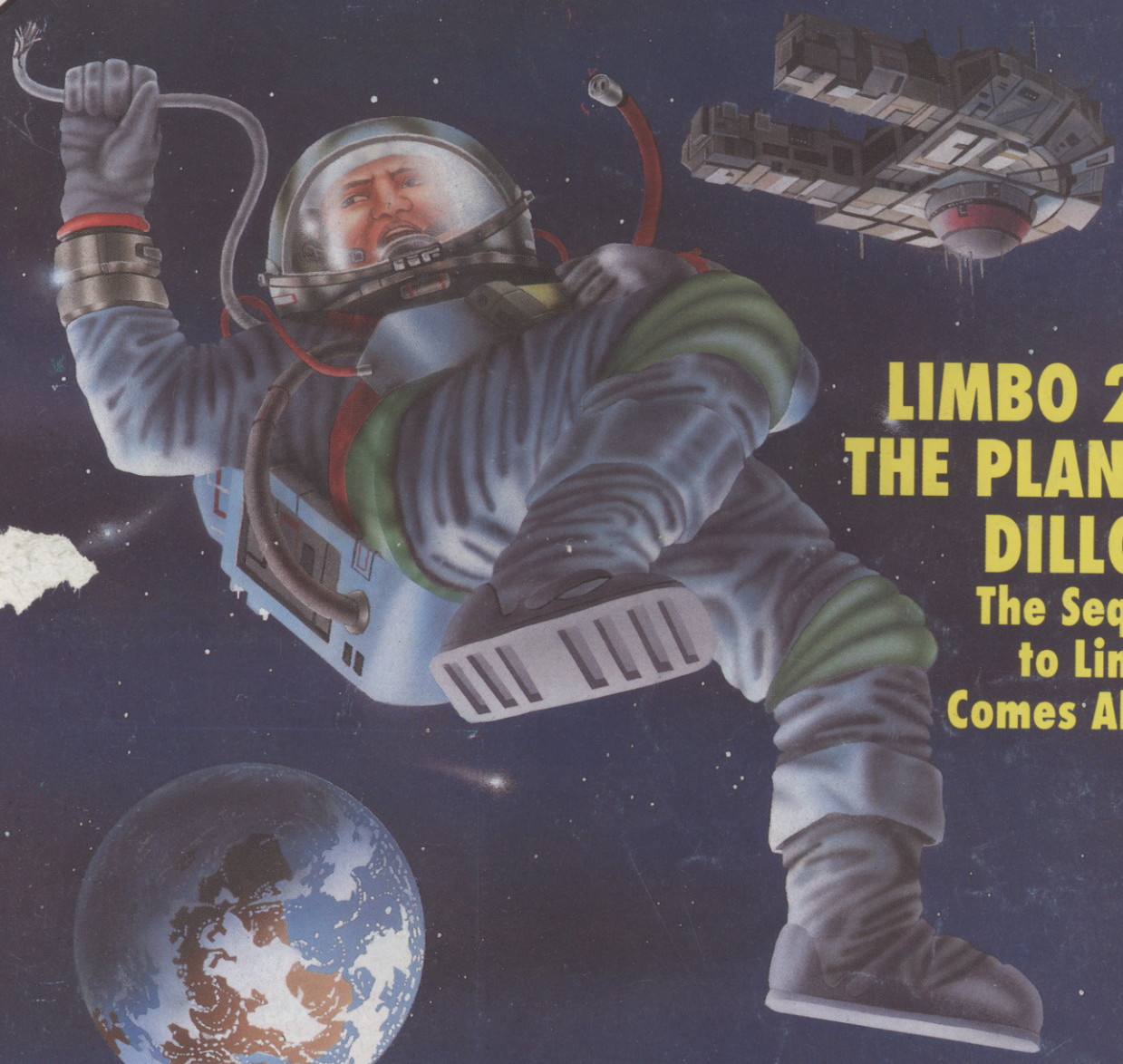
AUGUST 1990 £2.75

Commodore DISK USER



LIMBO 2 – THE PLANET DILLON

The Sequel to Limbo Comes Alive

- Techno-Info
- Geoprogrammer Reviewed
- Further Adventures in 'C'

9 770953 061007 08

ONE DAY AT SCHOOL...
OH NO! NOT MATHS AGAIN!
MATHS TEXTBOOK
AND MR JONES IS SUCH A ROTTER!!
YOU KNOW YOU TOLD ME ABOUT THE SCHOOL COMPUTER CLUB?
MATHS TEXTBOOK
YEAH?
WELL I RANG F.A.S.T. AND TOLD THEM ABOUT IT.
AND HOW MR JONES LETS EVERYBODY COPY SOFTWARE, AND EVEN DOES IT HIMSELF?
I'M NOT SURE WHAT THEY'LL DO, BUT THEY WERE INTERESTED
I HOPE THEY GET HIM...
THEN I'LL SHARE THE REWARD WITH YOU!!

£1000 REWARD

FOR INFORMATION LEADING TO A PROSECUTION & CONVICTION

THIS CAMPAIGN IS ORGANISED BY

ELSPA

EUROPEAN LEISURE SOFTWARE PUBLISHERS ASSOCIATION

ANY INFORMATION ON PIRACY SHOULD BE PASSED TO F.A.S.T. (THE FEDERATION AGAINST SOFTWARE THEFT)

TELEPHONE 071-497 8973

PIRACY IS THEFT

CONTENTS

Commodore DISK·USER·

Volume 3 Number 10 August 1990

IN THE MAGAZINE

Welcome
Editors comments and instructions 4

Geoprogrammer
We review this Geos assembler/debugger 6

Further Adventures in 'C'
More on programming in 'C' 17

Adventure Helpline
Some more help for Kron adventurers 21

CBM Development Assembler
We iron out some of the bugs 33

Techno Info
Our guru keeps sorting out the problems 38

Software Offer
An offer you cannot turn down 41

ON THE DISK

Limbo II
The sequel to Limbo gets you moving 8

Screen Designer 128
Screen designing made easy 12

Database 78
A database full of features 15

Letter Maker
Text screens made decidedly pleasing 19

Functions
Make full use of those function keys 20

Games List Creator
Keep tabs on your games disks 22

Dual Diskcopy
At last an intelligent disk copy program 23

Sequencer 64
Musicians have a field day 26

Security
Put all those broken joysticks to good use 30

Superboot!
Auto load your programs 32

Publisher: Hasnain Walji
Editor: Paul Eves
Consultant Editor: Stuart Cooke
Technical Assistant: Jason Finch
Advertisement Manager: Deborah Brennan
Designer: Helen Saunders
Distribution: S.M.Distribution
Printed By: Gibbons Barford Print

Subscription Rates

UK	£33.00
Europe	£39.00
Middle East	£39.30
Far East	£41.60
Rest of World	£39.70 or $69.00

Airmail rates on request
Contact: **Select Subscriptions.** Tel: (0442) 876661

Commodore Disk User is a monthly magazine published on the 3rd Friday of every month. Alphavite Publications Limited, 20, Potters Lane, Kiln Farm, Milton Keynes, MK11 3HF. Telephone: (0908) 569819 FAX: (0908) 260229. For advertising ring (0908) 569819

Opinions expressed in reviews are the opinions of the reviewers and not necessarily those of the magazine. While every effort is made to thoroughly check programs published we cannot be held responsible for any errors that do occur.

The contents of this publication including all articles, designs, drawings and programs and all copyright and other intellectual property rights therein belong to Alphavite Publications Limited. All rights conferred by the law of copyright and other intellectual property rights and by virtue of international copyright conventions are specifically reserved to Alphavite Publications Limited and any reproduction requires the prior written consent of the company

© 1990 ISSN 0953-0614

INSTRUCTIONS

EDITORS COMMENT

"Old magazines never die! They move on to new publishers"

A hearty welcome to one and all. Six weeks ago it appeared that my world, and that of thousands of other C64 users, had come to a sudden and dramatic end.

As you are all aware, Argus Specialist Publications decided, amid much controversy, to close down it's computer magazine operations. At the time the future of CDU, YC and Your Amiga was somewhat uncertain. I am very pleased to say that we are now continuing with our work thanks to the vision of the new publisher Mr. HASNAIN WALJI - Managing Director of ALPHAVITE PUBLICATIONS, UNIT 20, POTTERS LANE, KILN FARM, MILTON KEYNES, BUCKS, MK11 3HF - Telephone: 0908 569844, FAX 0908 260229. I for one, would like to take this opportunity of thanking Mr. WALJI for his extremely good business sense in realising what potential these magazines have for C64 users in general. Thank you HASNAIN.

This issue, believe it or not, has been put together over a period of 5 days instead of the usual three weeks, thanks to the marvel of D.T.P. (Desk Top Publishing if you did not already know).

With the added resources at our disposal, we can all look forward to an even better produced issue next time round.

Finally, I would personally like to thank all of you that have been concerned over my recent illness. I am glad to report that all operations have been carried out with great success, I am now back on my feet. Now on with this months issue.....

Although we do everything possible to ensure that CDU is compatible with all C64 and C128 computers, one point we must make clear is this. The use of 'Fast Loaders', 'Cartridges' or alternative operating systems such as 'Dolphin DOS', may not guarantee that your disk will function properly. If you experience problems and you have one of the above, then we suggest you disable them and use

BACK ISSUES

Back issues of CDU are available at £3.25 per issue, which includes postage and packing via: Select Subsription Ltd, 5, River Park Estate, Berkhamsted, Herts, HP4 1HL Telephone: 0442-876661

The following back issues were available at the time of going to press:

VOL 2 No.4 MAY/JUN '89

BASE ED - Get organised with this C64 database.
DBASE 128 - 40 or 80 column storage for C128 owners.
6510+ - The ultimate in C64 assemblers.
SID SEQUENCER - Make commodore music with ease.
LIBERTE - Escape the POW camp in this 1940's style adventure.
FX KIT - Bangs, Pows and Zaps made easy.

VOL 2 No.5 JUL/AUG '89

FONT FACTORY - Create your own characters.
HIRES DEMO KIT - Add music to your favourite picture.
ANIMATOR - Get those sprites moving
BORDER MESSAGE SCROLL - Say what you like along the bottom of the screen.
TYPIT-128 - Create professional text layout on your C128
SCREEN COPIES UTILITY - Download your favourite screens, including CDU paint files.
VIDI-BASIC - Graphic based extension to Basic.
64 NEWS DESK - Become a C64 reporter.

VOL 2 No.6 SEP/OCT '89

MICKMON - An extensive M/C monitor.
SCRAPBOOK - Collectors and hobbyists database.
CELLRATOR - Enter the caves if you dare.
RAINBOW CHASER - Rainbows means points in this unusual game.
HIDDEN GRAPHICS - Utilitse those graphic screens.
FORTRESS - Save the world! Yet again.
DISK HUNTER - Keep tabs on your disk library.
SUPERFILE - One more for the record keepers.

VOL 3 No.1 NOV '89

BASIC EXTENSION - Windows and Icons the easy way.
B-RAID - Vertical scrolling shoot'em up.
DISKONOMISER - Prudent disk block saving.
HELP - Design your own information help screens.
ORSITAL - An arcade style game with a difference.
PROGRAM COMPARE - Basic program development has never been easier.
RASTER ROUTINES - A few colourful demos.
SPRITE EDITOR 1 - A no nonesense basic sprite editor.
WABBIT - Help the rabbit collect his carrots.

VOL 3 No.3 JAN '90

4 IN A ROW - Connect a row of counters.
FROGS IN SPACE - Leap to safety across the space lanes.
BLACKJACK - Don't loose your shirt.
LORD OF DARKNESS - Defeat the evil lord in true adventure style.
MARGO - Fly around and collect jewels and fuel.
JETRACE 2000 - Have you got what it takes to be best?
ULTIMATE FONT EDITOR - Create your own screens, layouts and characters.
SELECTIVE COLOUR RESTORER - Design your own system start up colours.
6510+ UNASSEMBLER - Transofrm 6510+ M/C into source with labels.
TRIVIA CHALLENGE - The first of 3 files for this superb game.

VOL 3 No.4 FEB '90

COLOUR PICTURE PRINT - Download your favourite colour screens.
BASE-ED2 - An update to our popular database system.
1ST MILLION - Play the market in this strategy game.
FM-DOS - Enhance your drives operating system.
GEOS FONTS - A further 4 fonts for Geos users.
HASHING IT - Reletive file programming made easy.
MULTI-SPRITE - Make full use of up to 24 sprites.

the computer under normal, standard conditions. Getting the programs up and running should not present you with any difficulties, simply put your disk in the drive and enter the command.

```
LOAD"MENU",8,1
```

Once the disk menu has loaded you will be able to start any of the programs simply be selecting the desired one from the list. It is possible for some programs to alter the computers memory so that you will not be able to LOAD programs from the menu correctly until you reset the machine. We therefore suggest that you turn your computer off and then on again, before loading each program.

HOW TO COPY CDU FILES

You are welcome to make as many of your own copies of CDU programs as you want, as long as you do not pass them on to other people, or worse, sell them for profit. For people who want to make legitimate copies, we have provided a very simple machine code file copier. To use it, simply select the item FILE COPIER from the main menu. Instructions are presented on screen.

DISK FAILURE

If for any reason the disk with your copy of CDU will not work on your system then please carefully re-read the operating instructions in the magazine. If you still experience problems then:

1. If you are a subscriber, return it to:
   Select Subscriptions Ltd
   5, River Park Estate
   Berkhamsted
   Herts
   HP4 1HL
   Telephone; 0442-876661

2. If you bought it from a newsagents,
   then return it to:
   CDU Replacements
   Protoscan Europe PLC
   Burrell Road
   St. Ives
   CAMBS
   PE17 4LE
   Telephone; 0480-495520

Within eight weeks of publication date disks are replaced free.

After eight weeks a replacement disk can be supplied from Protoscan for a service charge of £1.00. Return the faulty disk with a cheque or postal order made out to PROTOSCAN and clearly state the issue of CDU that you require. No documentation will be supplied.

Please use appropriate packaging, cardboard stiffener at least, when returning disk. Do not send back your magazine, only the disk please.

NOTE: Do not send your disks back to the above address if its a program that does not appear to work. Only if the DISK is faulty. Program faults should be sent to: BUG FINDERS, CDU, Alphavite Publications Ltd, Unit 20, Potters Lane, Kiln Farm, Milton Keynes, MK11 3HF. Thank you.

DIRECTORIES EXPLAINED - Find your way round the directory jungle.
TRIVIA CHALLENGE - The 2nd part.

VOL 3 No.5 MAR '90

PLAGUE - Become your planets Guardian and Defender.
SURROUND - Reversi on the C64
GEOS FONTS - The last of 12 new Geos fonts.
SCREEN SLIDE - Create your own slideshows.
JOYSTICK TESTER - Put your stick(s) through the mill.
COLOUR MATCHER - Mastermind for the younger players.
SCREEN MANIPULATOR - Full use of the screen now obtainable.
VIDEO RECORD PLANNER - Keep tab on your home recordings.
TRIVIA CHALLENGE - The 3rd and final part of the game.

VOL 3 No.6 APR '90

BAR PROMPTS - M/C input routine.
HI-LITE BARS - Input routine but in Basic.
TEXAS DEMO - Example of using Basic in demos.
CHARS TO SPRITES - Convert UDG's to sprites.
FONT FACTORY - Complimentary program to the above.
3D-TEXT MACHINE - Impressive 3D text screens the easy way.
SCREEN ENHANCER - Makes full use of the screen easy to achieve.
SPREADSHEET 64 - An excellent, easy to use spreadsheet.
MINI-AID - 3 short utilities to aid the Basic programmer.
C128 COLLECTION - 3 very useful C128 programs.

VOL 3 No.7 MAY '90

NUDGE - FLD explained in laymans terms.
WINDOW WIPER - An alternative screen wipe system.
CHARACTER EXTRACTOR - Borrow those nice character sets you see.
MAZE GENERATOR - Create your own fun.
HIRES ANIMATOR - This difficult subject made easier.
SPRITE DRIVER - Platform game designing without the fuss.
ROTOTRON - Demonstration of Sprites and Sound.
TEXT COMPRESSION - How to squeeze a gallon into a pint.
SCREENS - Make up your own help screens and keep them in memory.
INTERRUPT POINTERS - Geos style windows and pointers for you.

VOL 3 No.8 JUN '90

ALEATORY MUSIC - An alternative music system.
SPRITE BASIC - Efficient sprite handling through Basic.
SPRITE GENERATOR - Another sprite editor for your library.
MUNCHER - Pacman returns with a vengeance.
ASTRODUS - Escape the spaceship Astrodus in this adventure.
1581 DIRECT ACCESS - Find your way around the 1581 disk drive.
PERSONAL ORGANISER - Design your own organiser pages.
128 CONVERTOR and MATHS AID - 2 more for C128 users.

VOL 3 No.9 JUL '90

QUICK MERGE 64/128 - Another useful routine for your archives.
THE GAME PLAN - An aid to knowing what where in your games.
CHARACTER dESIGNER - Another designer for those without.
HASHBASE 128 - A powerful database for C128 users.
REVASM 64/128 - Two unassemblers for non Speedy Assembler owners.
SPEEDY UNASSEMBLER - An unassembler specific to Speedy Assembler users.
BANKS AND MEMORY - An aid to redifing screen and graphic memory.
GRAPHICS FACTORY - A novel way of getting in graphic design.
POT POURRI - A selection of useful routines for all users.

All orders should be sent to:- Select Subsriptions Ltd, 5, River Park Estate, Berkhamsted, Herts, HP4 1HL. Please allow 28 days for delivery.

FEATURE

GEOPROGRAMMER

If you have ever wanted to produce your own GEOS environment applications this utility package is just for you

BRIAN SEDGEBEAR

I have always found that when programmers talk about assembly language whether it is on a C64 or Amiga, they usually make it out to be some sort of mystical art that only the super brains of the computer world can grasp. My motto is; have a go; so if I unintentionally make Geoprogrammer sound complex, it is only in the sense that it is sophisticated. Geoprogrammer is not designed to teach you assembly language but it makes things a little easier. With it, BERKELEY SOFTWORKS have shown their commitment to Geos and the support in helping users and third parties to develop their own application software operating within the Geos environment. But don't be mislead as Geoprogrammer will allow you to write your own stand alone programs which can be run completely independent of Geos.

Geoprogrammer is supplied in usual Berkeley packaging, consisting of a double sided disk and a 400 page manual which is written to the usual high standard. On the disk are the three main parts of the software, namely, Geoassembler, Geolinker and Geodebugger. It also contains files with the complete Geos operating system equates and macros and three sample Geos applications to show you just how simple it all is. Before we go any deeper, an equate is a constant or absolute memory address. A macro is a machine code routine that is given a name and can be inserted in any other machine code and can be called by simply inserting it's name. Macros can also be a set of keystorokes or commands called by a single name in Geodebugger.

Berkeley say that Geoprogrammer is a scaled down version of their UNIX based development system, used to produce the Geos operating system and it's applications. Perhaps a slight exaggeration, but it is a superbly designed development system and when used in conjunction with the Commodore RAM expansion units it provides one of the best debuggers I have seen on the Commodore 64. Introductions over, slight pause for sunlight, refreshments and fresh air, and we will take a look at a standard development session.

Geoassembler is designed to convert 6502 assembly language source code and produce a linkable object file. Simple enough but Geoassembler takes it's source text from Geowrite, which may seem a bit strange but does have it's advantages over many similar products, so our development must start with Geowrite. The Geowrite document can contain graphics and can also contain icon images, all of which will be automatically converted into binary data by Geoassembler. Also, thanks to the Geowrite environment, Italic or Bold type styles and different fonts can be included to highlight importan sections of the source code. Anything normally done in writing a document can be done in the source listing without any effects on the assembly process.

Because Geoassembler recognises and uses labels, the source code can have symbolic names making it much easier to understand at a later date. How many times have you written a piece of code only to come back to it a few months later and not have a clue what it was supposed to do? Also, it removes the necessity to remember important addresses and routines. Geoassembler even accepts and distinguishes between local and global equates as shown in fig 1.

It supports all official MOS technology mnemonic instructions, also, support is given to macros which can reduce the length of the source code and make it much simpler to understand by allowing a simple pre-defined command to replace a standard or common section of code. Using macros, a complete library of sections of code can be constructed in the same way many software

You might use a macro like this in your code.

```
SuBW subtrahend,minuend
                ;subtract word
```

Geoassembler could expand this into the following at the desired location automatically.

```
SuBW:
 lda  minuend    ;get byte value
 sec
 sbc  subtrahend ;subract low  byte
 sta  minuend    ;overwrite minuend
                  with result
 lda  minuend+1  ;high-byte with
                  carry
 sbc  subtrahend+1
 sta  minuend+1
```

In this case, SuBW is the label for the macro.

Figure 1

developers use standard blocks of code fitted together to produce a program. Fig 1 shows an example of this at work.

Geoassembler includes a complete mathematical evaluator and standard mathematical symbols can be used in the source code. These are handled at assembly time and also increases code readability.

Once the source code is written it can be saved to disk as a normal Geowrite document. The next stage is to load Geoassembler and assemble the source code into a relocatable object code file. This is not the final form as it has no specific address and true machine code. In this form, it could be located anywhere in the computers memory and still operate correctly once linked to that address. It is good practice to keep a copy of your routine in this form in case, once the entire program is complete, you need to relocate the routine.

The next stage is to use Geolinker to anchor the relocatable object code to an absolute address. Geolinker is also used to connect multiple relocatable object code files together to form a complete application that can occupy the complete computer memory. Geolinker is capable of producing Geos VLIR and Sequential files and also standard Commodore application files for running outside Geos. Geolinker uses a seperate file formed in a similar way to our original source code, using Geowrite to control the linking process and give directives, a file name and expression that affect the linking process. Geolinker is able to resolve cross-references involving different relocatable object code files, wow!. Sounds complicated but what this means is, if you have a label which is used throughout different parts of your final application and maybe different files, it need only be defined in one of the object code files and Geolinker will carry it across to the other sections. Perhaps the most impressive feature of Geos style programming is the VLIR file, which, typically has one main resident section and many overlay modules which are loaded in over the top of code that is no longer required, except when a new section is required. This means that theoretically, an application many times greater in size than the maximum computer memory can be produced and run. Geolinker will allow any user to design these powerful applications with only a few restrictions.

Finally we come to what is usually the most time consuming, but rewarding part of software development, debugging. (assuming it does not work first time of course). This is where the final part of Geoprogrammer comes into it's own, Geodebugger. At first, I was surprised to see that using Geodebugger, you leave the Geos environment and return to a dull plain text display, but the reason for this is to pack as many features into memory as possible.

At this point I found it vital to have a Commodore Ram Expansion unit (REU) in place, as the mini-debugger which Geodebugger is supposed to default to (but I discovered that it stubbornly refused to do, unless I manually force it) is very restrictive. Not surprising when you consider the memory restrictions imposed when running the Geos environment, Geodebugger and still leaving enough room for an application. With an REU, Geodebugger loads the super-debugger into the expansion Ram leaving the standard computer memory almost untouched, providing a heck of a lot more features.

Differences aside, you can run your application along side Geodebugger and interupt it at any time to alter, single step, set breakpoints, check registers and disassemble any code. It includes a complete expression evaluator and is therefore capable of disassembling and labelling the machine code with your own original labels. It even has a complete macro language. I was disappointed to find that there is no printer support in Geodebugger, therefore you cannot make a hard copy of a section of code to take away and study out in the sun when bug hunting. I believe this is a bad omission and cannot see why it was not provided, expecially when all other Geos applications have such good printer support.

All in all, a very desireable purchase for any aspiring Geos programmer or C64 programmer for that matter, also a very useful utility if you are learning to program in assembly language. I would like to see any future versions include better printer support, and would advise any serious programmer to invest in a Commodore memory expansion unit to make full benefit of the software.

It would be nice to see distributors supplying Geoprogrammer packaged with the REU, or Geos packaged with the REU with a reduction in price as we have seen with disk drives.

I hope that I did not get too involved, although I could have mentioned many other features, but I hope that I have provided useful reading for Geos users. If I had just struggled writing a game using a small memory monitor I would be feeling seriously sick.

Suppliers: Financial Systems Software Ltd, Masons Rhyde, Defford Road, Pershore, Worcs
Price: £39.95

ON THE DISK

LIMBO II - Planet Dillon

Like all good movies, computer games have sequels to the original story. CDU presents for your pleasure LIMBO II - Planet Dillon

Many of you will recall LIMBO that was on the DECEMBER 1989 disk of CDU. Unfortunately, as you may now realise, our intrepid Editor made one very tiny error at the time. He put the wrong version of the game on the disk, consequently the one published had a few problems built in. (Otherwise known as bugs...!!)

We now present for your entertainment and amusement the updated version of the original, aptly titled LIMBO II - Planet Dillon. As before, plug your joystick into port 2 to move Wolthamstow around each zone. The idea of the game is to clear all of the blobs off the squares whilst avoiding the Dillonite guards. Contact with these or the scrolling background results in energy loss. Flashing squares will electrocute you. Blocks with moving squares on them are random squares. If you move onto one of these and press the fire button, something at random will happen. (It could be good, bad or extremely bad). Squares with moving arrows on them hurl you off in that direction. That's all there is to it!!

The Editor in one of his more laid back moments decided that it would be a good idea, if every now and then the readers got to know a little about the programmers that provide us with our entertainment. So here, for your inquisitive minds, is a profile on the author of LIMBO, Steven Pattullo. Take it away Steve.

STEVEN PATTULLO

FULL NAME: Ste Pattullo
AGE: 20
HEIGHT: 6'1"
PREVIOUS GAMES: Albert saves the world again. Albert saves the world again II. Hyperactive. UFO. UFO II. UFO III. Sphere. Sphere II. Trivial Challenge. Limbo. Platformania and numerous articles on programming.
FAVOURITE FOOD: Anything that's classed as unhealthy.
WORST FOOD: Vegetables.
FAVOURITE DRINK: Lager (Holstein Pils on draught).
PETS: One extremely fat Cocker Spaniel, 2 fish and a younger brother.
HOBBIES: Drinking and playing snooker.
FAVOURITE GAMES: This is going back a long way to my Spectrum days. I would have to go for Manic Miner, Jet Set Willy and a game that was never released called Pud Pud.
WORST GAMES: A lot of the stuff that is being churned out now. The problem is that people are trying to make games better and better and more often than not, you end up with a game that is only suitable for someone with an IQ of 6 million and has 2 dozen pair of hands. I think people should go back to the formats of about 1984-1985. The games that were around in this period may not have had the best graphics and sound but they were a damn sight more playable the a lot of stuff these days. (I rest my case) {I must admit that I agree with you Ste...Ed!!}
FAVOURITE SOFTWARE AUTHOR: Matthew Smith (when he was around). If he wrote a game today I would go out and buy a Spectrum.

Thank you Steve for imparting such wisdom to us. I think as compensation for putting the wrong game on the disk originally, you can now give us your diary of events leading up to LIMBO II being finished. Take it away (again) Steve.

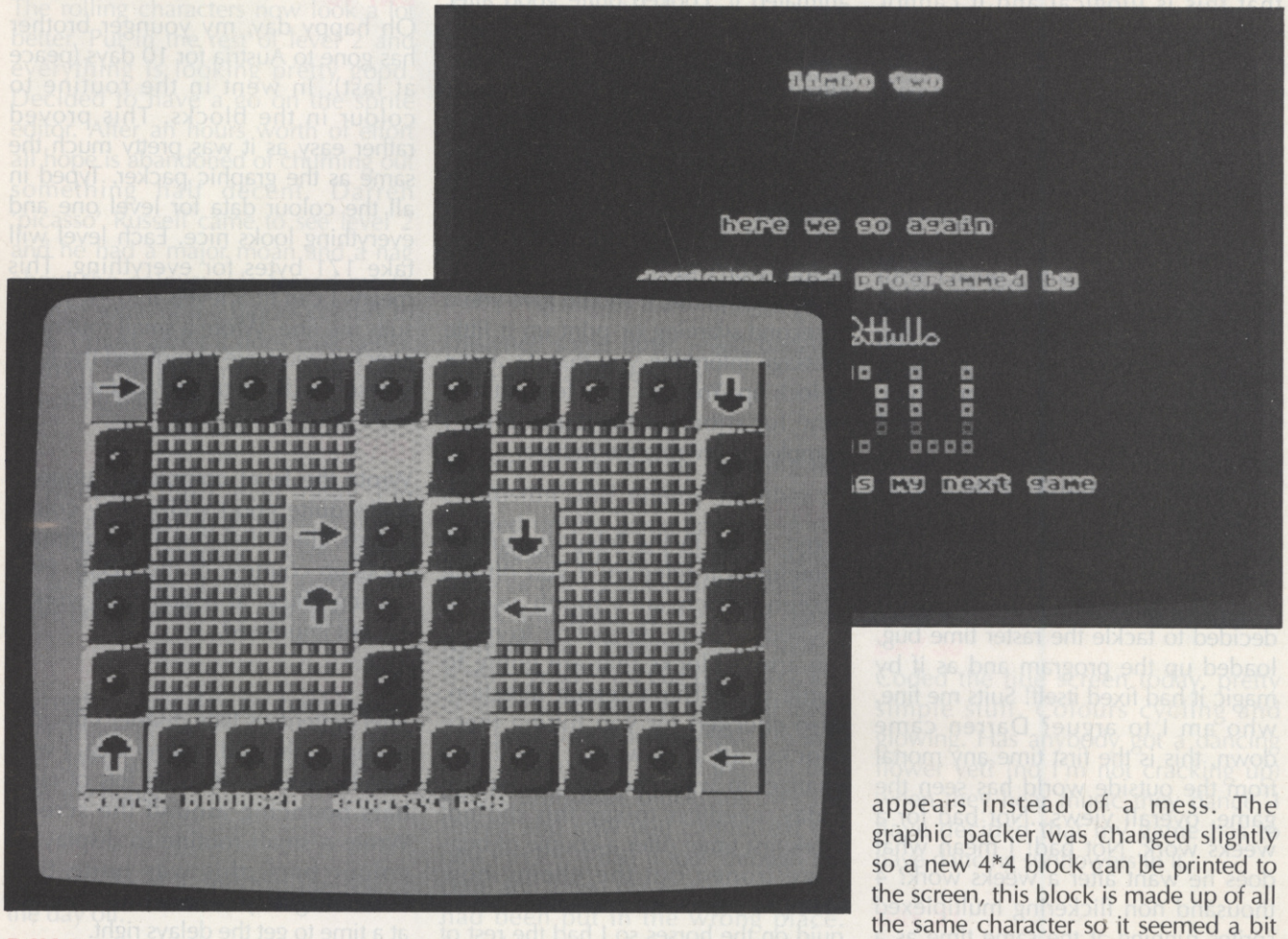
DAY 1

CDU have asked me to write a diary for my new game (fame at last). The game is going to be a follow up to one of my other games "LIMBO". So far I have designed a few graphics but nothing to write home about. Spend the rest of the day thinking about the game plan, various things spring to mind such as:

1. Is it going to scroll?
2. What style of graphics should I use?
3. What type of sprites are going to be used?
4. What colour schemes should I use?

The answer to all these questions is....erm...I don't yet know. Tried to think of a name for the game, fail, some intelligent person in the pub suggested LIMBO II, what an imagination.

ON THE DISK



DAY 2

Decided that the game will be a static screen and the graphic blocks will be designed so that they can be any one of 8 colours plus the 2 multicolours. Wrote the graphic packer, this packs the screen into 60 bytes instead of 1024, quite a saving me thinks. Fed some data into the packer in hope of 60 blocks gracing the screen, instead a mass of junk appeared, (that's a good start), had a look through the code, fixed a few bugs, ran the program again and this time got a blank screen. It was at this time that I decided that it would work better with a character set in memory. Wrote a small routine to animate the dots in the middle of the blocks, this worked first time. Flushed with success I put the main sprite on the screen, Uurrrghhhh! In my infinite wisdom I designed the sprite on a black background forgetting that the game is played on a light play area. Changed the sprite and the colours and tried again. Looks much better with a black outline around it. Decided to stop work while the going's good.

DAY 3

Off to Liverpool to get my Amiga out of hospital. At the shop I was informed that it needed a new drive, new mouse, and several new chips and circuit boards. I wonder if you can get Amigas into BUPA? Messed around with my revamped Amiga for the rest of the day.

DAY 4

Wrote the joystick routine and put in the screen boundaries. You can now move the character happily around the screen. Decided it was time you could eat the dots so in went the routine to check what character was under the sprite. If the character is part of the dot then remove it from the screen, simple. This routine will be used to check all of the other blocks you can touch.

DAY 5

Altered the pick up dot routine so that when you pick up a dot a hole appears instead of a mess. The graphic packer was changed slightly so a new 4*4 block can be printed to the screen, this block is made up of all the same character so it seemed a bit of a wasted having sixteen characters the same, and anyway this block will be rolled around. Wrote the routine to roll the characters in this block, the block now rolls in the opposite way to the way that the joystick is pointing, hmmmm looks ok but not too good. I'll keep it there for now anyway. Had a good idea on how to move the enemy around the levels, I'll try that tomorrow.

DAY 6

Hopefully the enemy movement routine will be put in today. The way I moved the enemy in LIMBO was to read in the data for direction, speed and how far the sprite should travel before switching course. This took up too much data for my liking, so in this game the enemy will follow arrows around each screen. Wrote the routine to check this for the first sprite and assembled it in great confidence.....well an enemy sprite is there but he only moves jerkily when I move the joystick. I have decided

that this is illogical and it cannot possibly happen. After a good look through the code I discovered the bug, the program was jumping past my sprite move routine unless you moved the joystick (surprise surprise) Altered the code so it does access my sprite routine and low and behold Mr. Nasty sprite starts to zoom around the screen, he comes into contact with one of my down arrows and goes up. This is not good. Fixed this small bug and he now trundles off in the right direction which is jolly decent of him. Had a look at how much raster time I was using, wish I hadn't bothered, my latest ultra compact sprite mover routine takes about half a screen full. After a long think I decide there's only one thing left to do PANIC!!!!

After a quick pint in the Bradley, I decided to tackle the raster time bug, loaded up the program and as if by magic it had fixed itself! Suits me fine, who am I to argue? Darren came down, this is the first time any mortal from the outside world has seen the game, overall views...Not bad for a weeks work. Not bad! I mean what does he want after a weeks work? 4 thousand non flickering multiplexed sprites running at the same time as a 72 piece digitised orchestra and a sampled after dinner speech by the Queen??? He thought this might improve it a bit.

DAY 7

Put all seven enemy sprites on the screen and they follow each other happily around, they even obey my direction arrows. Oozing with confidence a couple more direction arrows are thrown onto the zone and they totally ignore these. Hacked around with the character detection routine for a bit and everything now works ok. A few more character blocks were designed for nasty things, such as arrows which push you along in one direction. The game now has a name courtesy of Julian Dolan, he came up with 'The Planet Dillon' which is really stupid but who cares? That should do it for today.

DAY 8

Designed the teleport block and animated it. Looked quite good after 10 efforts on the character editor. I have decided that the enemy will be allowed to teleport as well as you. I mean just because they're enemy they don't have to have all their privelages taken away. Wrote the code to teleport you...assembled it....and CRASH!! Arrrggghhhhhhh!!! Even restore won't save me this time. This really annoys me, it should give you a 5 second warning or print up a little message informing you that it's about to lock up. You can now teleport to a new location on the zone after I completely rewrote the routine.

DAY 9

Fixed a bug in the teleport routine, sometimes it teleported you off the screen which isn't much use to anybody. Wrote the routine to move the arrow blocks that are going to hurtle you off in a certain direction. Also checked for you hitting these blocks, all this works fine for a change. In went the sprite animator today, needless to say this did not work first time. After a couple of bugs were ironed out the enemy now stamp around the screen. Won 75 quid on the horses so I had the rest of the day off.

DAY 10

Put level one in today and worked out how many bytes I would need per level to set up colours and animation etc. Decided to put in an exploding block if you touch it. It explodes underneath you so I will have to use a sprite for this, it would take up too many characters otherwise.

DAY 11

Today saw the introduction of the level selector, this routine works out what level you are on and gets the relevant data to set up the sprites etc. Level one is now playable I am pleased to say.

DAY 12

No work done today but Desmond Rigby came round and insisted I put his name in the diary, Hello Dez....Happy now?

DAY 13

Oh happy day, my younger brother has gone to Austria for 10 days (peace at last). In went in the routine to colour in the blocks. This proved rather easy as it was pretty much the same as the graphic packer. Typed in all the colour data for level one and everything looks nice. Each level will take 171 bytes for everything. This means I can fit 5 levels into 1K which suits me fine. Wrote a small routine to check if you have completed the level and sent a demo off to Paul Eves.

DAY 14

Wrote the level complete routine, it looks quite good with a few colour cycles, then disaster. The computer locks up courtesy of the CEGB. Even hurling abuse at it does not work. I am not impressed.

DAY 15

After yesterdays little disaster I decided to change the end of level idea, I want it to be like LIMBO with lots of raster bars bouncing around the screen. Wrote a small routine to display 50 colour bars, this was quite time consuming as you have to do one bar at a time to get the delays right.

DAY 16

Started to put some different colour bars into the table and arghhh glitch, glitch, flicker, flicker. Disaster, all the bars have distorted. Fixed the glitching but it still looks terrible so out it went. I'll have to think of something else now.....thought of something else and done it. Much better, it looks a bit like the one from LIMBO, well it looks extremely similar to the one from LIMBO. Ok, I admit it, it IS the one from LIMBO, but who cares, I don't.

DAY 17

Put most of level 2 in today, it's really boring typing in loads of numbers but it's got to be done, finished this off and went to the pub.

DAY 18

Started off the day by sorting out my disks, I still haven't a clue where everything is. Found a bug in the rolling character routine and fixed it.

The rolling characters now look a lot better. Put in the rest of level 2 and everything is looking pretty good. Decided to have a go on the sprite editor. After an hours worth of effort all hope is abandoned of churning out something half decent. Darren 'picasso' Russell came to see level 2 and he had a major moan and a nag about the colours. He forced me to change some of the blocks colours to blue, I must admit it does look better. Darren looks smug.....consider punching him in the nose.

DAY 19

In went the third level today, I'm getting really fed up of typing in all this data and I've only done 3 levels. Wrote the score routine, it doesn't just add the score on, it counts it on Ooohhh.

DAY 20

Started off the day by backing up everything in sight onto a new disk and drew some new graphics which took me hours.

DAY 21

Typed in level 4 and had the rest of the day off.

DAY 22

Oh disaster, the level I saved out yesterday has turned into a sequencial file so I can't load it back into memory. Tried to rescue the file but no luck, I'll just have to type it all in again. This is a bad day, I've just received a letter (death threat) from the inland revenue and I can't find my accounts book anywhere. Re-typed in level 4 which was extremely exciting, I then managed to send a command down my serial port which makes the disk think that it is write protected (not good).

DAY 23

Designed levels 6, 7 and 8. I then found out that my enterprise allowance scheme has expired so I went into a mass panic.

DAY 24

Job hunting!!

DAY 25

Typed in all the data for the latest three levels which was immense fun. I decided it was about time the exploding block routine went in, so in it went. All that needs doing to it is the sprites designed for the explosion. I have decided that the teleporter will no longer be a teleporter but a random block that does different things when you touch it.

DAY 26

Designed some other levels because that's all I had time to do.

DAY 27

Started off the day by drawing the explosion and enemy number 2. Loaded up the game and put in the new sprites, not bad I suppose. Started some of the routines for the random block and some of them are really nasty. (ha ha, I think the strobe routine is really good). First bug of the day, lots of little blocks pop up here and there, very pretty but totally useless. I'm in a no win situation, I fixed the bug that kindly distributes blocks all over the screen and now you cannot eat the dots HELP!!! I had a look through the code and found out that one little line had been put in the wrong place. Everything is now going ok, so I wrote a small program to blank out the random block when you use it.

DAY 28

Had to re-write a small part of the random block routine because one of the files on the disk was adamant that it wasn't going to load. More work on the random block, it now has a lot more features on it. Designed and keyed in all of the data for level 8, put in the routine that counts down your energy and CRAAASSSHHH! Good job I saved out the level data before. I haven't lost much work really. Re-wrote the routines and saved out before running them, the same thing happens again even though it can't possibly. Ha! found it at last. I was thinking there was 4 bytes in the energy, in fact there are only 3.

DAY 29

Had a discussion with Darren last night about the game and he decided he didn't like the exploding block. He insists that is should electrocute you instead of blowing you up. This is equally violent but it saves me a sprite (ponder, what can I do with another sprite?). So in went the electrify routine, it's ok I suppose. Found yet another thing that needed altering, when you activate a random block it is replaced with background characters that take energy off you, so me being the generous type I replaced it with the dead dot graphics. I have decided that when getting electrocuted, the sprite will change so that he/she? looks unhappy. Well would you look pleased with 20,000 volts going through you? I think not. Designed the unhappy character and in he went. Made the random block more random and added a new feature. You can now die when you run out of energy.

DAY 30

Coded the title screen today, pretty simple stuff. Colours cycling and glowing. Has anybody got a dancing flower yet? (no I'm not cracking up) when they hear music they dance. I have decided that my game has to have dancing sunflowers in it.

DAY 31

Designed some graphics for a dancing sunflower and put it in the bottom border of the game. Designed another couple of levels and put them in. The game is now finished I just need some other things like game over.

DAY 32

Wrote the game over routine today, it looks quite effective with characters rolling around at different speeds, also put another thing in for those of you that dare press the RESTORE key???

DAY 33 (The final day)

Linked all of the routines up today and everything is working just fine. Put the game on a disk and off it went to Paul Eves. Well all you people in reader land, that's it, finito end of game. Hope you like it anyway.

Thank you Ste for a most informative and interesting diary of events. I certainly enjoyed playing your latest offering. I hope everyone else does also. Ed!

ON THE DISK

SCREEN DESIGNER 128

A handy utility to help in designing text screens in both 40 and 80 column mode

D.H.FABER

I'm sure you agree that a nice layout of menu and help screens makes any program more user-friendly. In designing these screens the C64 programmer can choose from a plethora of available programming aids, the 128 user however has very little software available to them. The utilities described here offer C128 programmer; an easy and familiar editor to design text screens (40/80 columns). The facility to save the screens in packed form, optionally with a loader which enables you to recall them with lightning speed in your own programs. Plus a software switch to turn the 80 column screen on/off during its raster blanking period.

You will need the programs SCREEN DESIGNER, SCREENLD CREATOR and V-SWITCH CREATOR on disk or tape. (All these programs are provided on the CDU disk).

USING THE DESIGNER

Assuming your C128 is in power-up state. Start the program by the command:- RUN"SCREEN DESIGNER". Three remarks are in order;

The screen type, 40 or 80 column, on which you execute the RUN instruction will also be the one you'll be working on. If you want to load or save a screen, the program uses the device which was used most recently before the program was started. Any option which would destroy the contents of the screen you have been working on is executed only after you have acknowledged an "are you sure?" question. The options available to you on the menu screen are;

F1 TO ENTER THE WORK PAGE.

If you start from scratch you will see a grid, which should help you to place the characters on the desired position, with the cursor flashing in the top left corner. The editor used should be familiar as it is the same one as used by Basic. In other words, you have the same facilities as in direct mode with all CTRL and ESC codes available. By pressing RETURN you leave the workpage and return to the menu. To avoid lines scrolling off the screen, scrolling is disabled when you enter the work page (ESC L enables it again). A few remarks and warnings;

Be carefull with deleting and inserting lines using ESC D and ESC I. For example, if you insert one blank line then try to delete it again you will probably discover you have deleted the next line as well. This happens since the editor was originally written as a line editor to input Basic statements. Do not accidentally press CLR, it immediately destroys your work, no questions asked. You may not switch to the other screen (ESC X) and if you do by accident, switch back immediately. Except for this ommission the program is, to my knowledge, reasonably idiot proof. Once on the workpage, you have one or more additional options available. F6 changes the background colour of the screen. In 80 column only, F7 and F8 toggle the flash and underline bit of the current attribute respectively. (And move the cursor by one position afterwards).

F2 TO PUT A NEW GRID ON THE WORK PAGE.

This option returns the workpage to the initial situation.

F3 TO LOAD A SCREEN.

A file is loaded from the same device as the screen designer was loaded from. It is irrelevant if the file starts with a loader or not (see below) but it should be of the correct type. (40 or 80 column). Following a succesfull load you will find yourself on the workpage.

F4 TO SAVE A SCREEN.

First you are prompted with the question whether or not you want to change the colour of spaces and shifted spaces. Since not only the characters but also their colours (as well as the background colour and, for the 40 column screen, the chosen character set) are stored, a judicious choice here allows the pack routine to produce more compact files. This option, however, also serves a purpose of my own. If you are familiar with my Mouse80 utility (YC July/August 1989) you may recall that the mouses' arrow assumes the colour of the characters it is moving over, instead of imposing its colour as the normal cursor does. By carefully using shifted spaces and normal spaces and defining different colours for each, you can have different colours for the arrow in different sections of a menu screen. The next question you have to answer is whether you wish to save the screen with or without a loader. The ins and outs of this matter will be treated in the next section. For the time being save an as yet incomplete

scren without loader. Finally, you have to define a file name and the program saves the screen on the same device as the screen designer itself was loaded from.

F5 QUIT PROGRAM.

When this option is selected a system reset is performed.

IN YOUR OWN PROGRAMS

Let us assume you have saved a screen with loader. The loader is a piece of code written in such a way that it is relocatable. To you this means that you can load the file to almost any address in RAM0 and execute it from there (of course you may also make it a part of your own programs). The only forbidden locations are the I/O area ($D000-$E000); no part of the file may be in the ROM underneath it; also you should be carefull when using low addresses which might be used by the operating system or by the Basic interpreter. In order for the loader to do its job there is a certain protocol to be followed. (Described here for Basic programmers, assembler fans will find it easy to translate to their own requirements).

Three pairs of zero page locations should be poked with addresses (low/high) as follows:

$FA/$FB (250/251) where to put the characters
$FC/$FD (252/253) where to put colour/attributes
$FE/$FF (254/255) start address of the loader

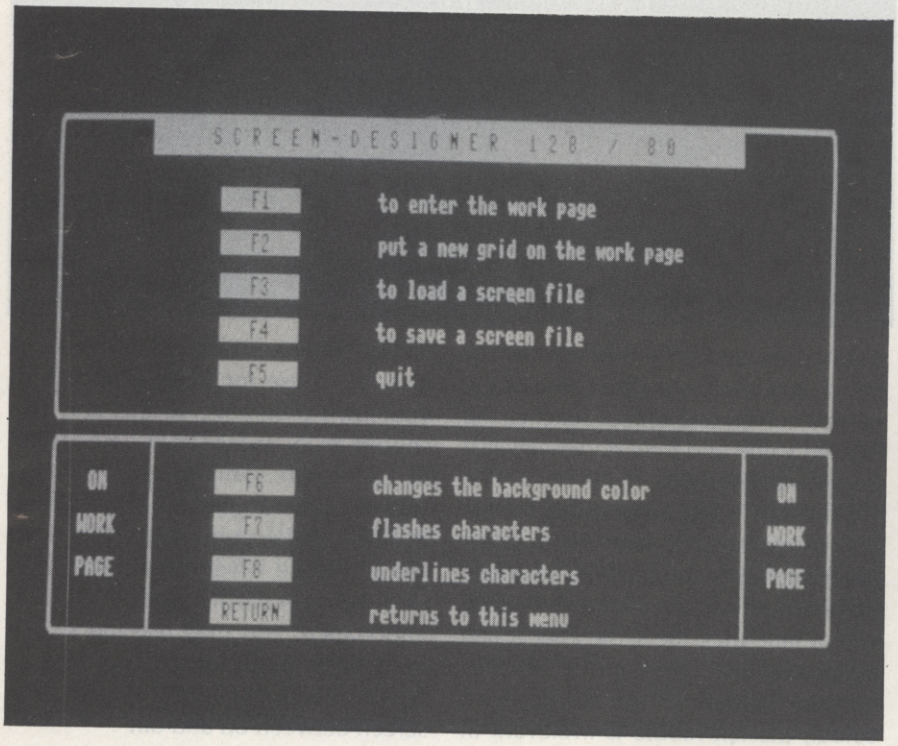
For example, if you have loaded a 40 column file to $9123 (37155) and the characters should go to the standard VICII text screen ($0400) and the colours to the colour ram ($D800), you should use:

```
POKE250,0:POKE251,4:POKE252,,0:
POKE253,216:POKE254,35:POKE255
,145
```

Next, you can have the screen shown by BANK0:SYS37155

For 80 column files the addresses specified in $FA/$FB (250/251) and $FC/$FD (252/253) pertain to the VDC's video ram, normally $0000 for the characters and $0800 for the attributes. This procedure works in direct mode as well, usefull for testing your screens.

The proper idea is not to unpack the file to the current screen, but rather to another location in memory and set the screen pointers to these locataions afterwards. It is your own responsibility to ensure that you do not try to 'unpack' and 80 column files to the 40 column screen and vice versa. Also, in 40 column mode you must take care that the packed file does not overwrite itself. The method described here is fine if you have only a few screens to show, else it is a waste to attach a 388 byte loader to each screen. A better alternative is described next.


SCREEN-DESIGNER 128 / 80
F1 to enter the work page
F2 put a new grid on the work page
F3 to load a screen file
F4 to save a screen file
F5 quit
ON WORK PAGE
F6 changes the background color
F7 flashes characters
F8 underlines characters
RETURN returns to this menu
ON WORK PAGE

AN ALTERNATIVE LOADER

When you are using any menu or help screens it is much more efficient to use a single loader and have the packed screens at various other locations in memory. To this end the designer can save screens without loader (if you should have forgotten if a certain file included a loader or not, or for which screen type it was meant, read the next section). This section describes such a loader for the 80 column screen, if you want to write one yourself for the 40 comumn screen, I again refer you to the next section. First let me enlarge upon the question of relocatability. Most utilities and programming aids are written in such a way that they should be loaded to a specific memory location. To the user this means a loss of flexibility. He/She cannot use this area for their own purposes or use other utilities that require the same memory locations. When dealing with machine code programs one might be tempted to publish assembler source listings instead. This is fine if everyone used the same assembler, but we do not. One way round this problem is to try and write completely relocatable code, containing no absolute references to memory locations within the program. If at all possible, this tends to make the programs longer than necessary and is feasible only for relatively small pieces of code.

For the alternative screen loader to be described in this section I opted for a different approach. The program SCREENLD CREATOR, which can be started by RUN<name> [ON Ux] in 128 mode, contains apart from the machine code, a table with addresses to be relocated. (Much as a linkage-

editor does in larger computer systems). After you have defined the base address the program saves the correct code to the same device as it itself was loaded from. In this way you can produce code for the location that is most convenient to you. The previous screen loader required you to define the target addresses for characters and attributes. This on retrieves these from the VDC registers and blanks the screen untill the new screen is complete. Also, although the loader itself must be in RAM0 the packed screen files may be located anywhere. Here's how to use it.

Poke addresses $FE and $FF (254/255) with the low/high address bytes of the screen file, then call the loader as 'BANK0:SYSxxxx,B' where B is 0 or 1 for a screen file in RAM0 and RAM1 respectively. For example, if the loader is at $9123 (37155) and the screen file at $A000 (40960) in RAM1 you would use:

```
POKE254,0:POKE255,160:BANK0:SY
S37155,1
```

WRITING YOUR OWN LOADER

If you want to write your own screen loader it is essential to know the structure of a screen file. First here's how to distinguish between a file with, and a file without a loader: the loader starts with a $08 byte (PHP), a file without never does. Not counting the optional 388 byte loader the file starts with a code byte with the following meaning:

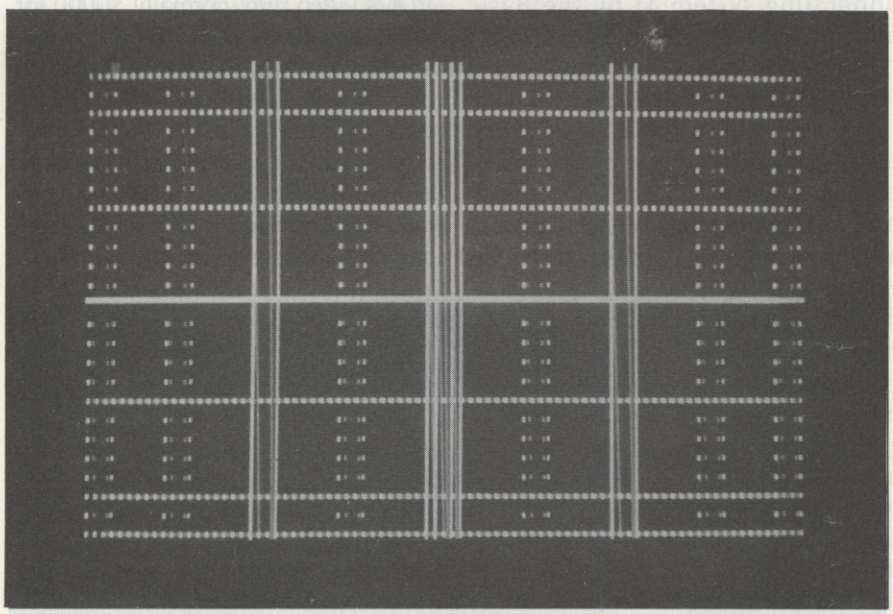
bit7: 0 or 1 for 40/80 column respectively
bit6: 0 or 1 chooses between character sets (40 column screen only).
bit5: Irrelevant
bit4: 1 (Thus this byte is never $08).
bits0-3 Define the background colour to be poked to $D020 and $D021 for the 40 column screen, and to the right nibble of VDC's register 26 for the 80 column screen.

Next follows two times 1000 (40 column) or two times 2000 (80 column) bytes, the characters and colour or attributes respectively. The pack routine used is a simple one, it only caters for strings of bytes recurring three or more times in a row and not for recurring patterns of bytes. Here is how to unpack the file.

First read a count byte, bits 0-6 make up a number between 0 and 127. If bit 7 was zero, the next (0-127) +1 bytes should be transported to the target address. If bit 7 was 1 then the next byte is to be duplicated (0-127) +1 times to the target address.

VDC SWITCH

f you experiment with the 2nd loader described above you will probably notice it neatly blanks the screen until the new screen can be turned on. When dealing with the VICII chip, which handles the 40 column screen, you have various options to achieve this effect. For example, you may use a raster interrupt outside the visible screen area to redefine the screen pointers or to turn the screen on or off; or you could read the value of the current screen line until it reaches the desired value and then do whatever it is you wanted to do.



The VDC chip, responsible for the 80 column screen, does not support raster interrupts nor can it be switched off (that is, by software). Nevertheless, with a little trick the same effecgt can be achieved. The right nibble of register 26 defines the background colour, the left nibble is irrelevant in multicolour mode (as used here), in monochrome mode it defines the foreground colour. Therefore, if the two nibbles are made equal then switching between multicolour and monochrome mode is equivalent to switching between screen on and off. Remains the problem of waiting for a "dark" period. You will know that the VDC is addressable only through a gateway of 2 bytes in the I/O area, $D600/$D601. When reading location $D600 bit 5 is 0 if the beam is writing part of the screen and 1 if it is in the blanking period. You have no means however to know how many clock periods the beam is away from entering the visible area again. Therefore the proper procedure is:

Wait for the beam to be on-screen. Wait for the next blanking period. This way you have sufficient time to change the contents of the registers. This method is used in the alternative loader described above, but it could also be of use for other purposes in your own programs. To this end you should RUN"V-SWITCH CREATOR" in 128 mode. This program allows you to create a machine code file for the base address you select. Since the piece of code is very small, 86 bytes, this relocatability is of special importance. You may use the switch from Basic as BANK0:SYS<address> where each call toggles the VDC screen between on and off.

DATABASE 78

A RAM based disk filing system which is simple to use and understand for all ages

C.F.SMITH

Many people that use computers do so not simply as a leisure activity, but also for some serious work. The most widespread use of any computer is as a filing system, either for large multinationals, small businesses or simply for the man in the street to keep track of his records. DATABASE 78 is a program that will enable you to do just that. It is a simple program that is user friendly and easy for people of all ages to understand.

One of the first things you will prompted for on running the program, is to state what type of printer you will be using, either CBM or STAR LC10-C. The name of the file that is present, if there is one, is displayed at the top of the main menu. From the main menu the following options are available.

A - START A NEW FILE

This option is self explanatory. It is used to create a completely new file from the begining. The maximum number of records is fixed at 300 and the number of fields is fixed at 4. The top of the screen states that you can enter a maximum of 78 characters in each field. The display as you type it in is how it will appear in the other options and on the printer. The file will not accept a nil entry in a field, so you will have to put something and I suggest you use a 0. Do not use commas or colons as any text entered after these will be lost. Typing EXIT to start a new field and then RETURN will delete that record and return you to the main menu.

B - LOAD or UNLOAD FILE

Follow the prompts and use this option to LOAD an existing file from your disk. If a file is already loaded then use it to clear the file before you load another.

C - SAVE THE FILE

Use this facility to save a new file that has not been SAVEd before. It can also be used to provide a safety or back-up copy on a seperation disk from your working disk. In the latter case use this last before switching off.

D - DISPLAY FILE (scroll)

This displays the file one record at a time. You can use the U and D keys to move UP and DOWN through the file. The number of each record is displayed at the top to provide an indication of your position in the file. The space bar takes you back to the main menu.

E - ADD RECORDS

This returns you to the point at which you left off when making your file originally. You are also told the record number you are working on. Don't use commas or colons as all text entered after these will be lost. Typing "EXIT" will return you to the main menu.

F - ALTER RECORD

You are first of all asked to name the record by its first (main) field. The display shows you the record as it exists and askes you which field you wish to operate on. This question is then replaced by the field that you state. After the field has been altered, keying RETURN will enter the change and ask you if you want to alter another. Keying "Y" will repeat the process and keying "N" will return you to the main menu.

G - SORT BY ANY FIELD

You are first asked for the field to work on. For example, if you enter DATE OF BIRTH as the name of one of the fields, it will arrange all the records in numeric order. If you type OCCUPATION, it will sort all the records of that field into alphabetical order. This is best appreciated by experimenting and then choosing option "D". If you have asked it to sort by the main field you can use option "V" in the SUB MENU to get a good idea of the result.

H - SUB MENU (sort and search fields)

This option takes you to a whole range of further options, including the PRINTER option. However, you can only get to the sub-menu if you have a file already present in memory.

I - DISPLAY RECORD (delete option)

You are first asked to name the record by its main field. The display allows you to view the record as a seperate entity but gives you only two choices, to delete or keep the record in memory. If you accidently delete a record from memory, all is not lost as you should still have it on disk, so long as you have not used option "J".

J - UPDATE DISK FILE

This option will maintain your running, or working, records up to date. It automatically deletes your old file of the same name from the disk and replaces it with your updated version, again giving it the same name. Anything you have deleted from that file has now gone, and anything you have added, deleted or changed will now appear or not at a later re-load. This option only works of course when you have LOADed an existing file from your disk. It should be the last option you choose if you have made any changes to your records, except for option "C", when you may want to make a back-up copy on a seperate disk.

K - DELETE FILE FROM DISK

This option will delete or scratch any file that you name from your disk. Be careful how you use this option. An escape facility is provided should you make an error.

L - EXIT PROGRAM

If you choose this option, you are reminded to update your disk file and

a chance of returning to the main menu is provided. If you continue the screen clears and the READY prompt appears.

Z - DIRECTORY DISPLAY

This option will display the contents of the directory of any disk without losing your loaded file. It will also indicate any protected files, which you can alter to suit your needs.

Database 78 keeps track of how many records there are present in the loaded file. At the bottom of the main menu this number is displayed for your information.

SUB MENU OPTIONS

M <> $

This is the first of a choice of five SEARCH and SHRINK routines that are available. By following the prompts and entering the data that you want omitted, a sub file will be produced made up of records that excludes that data in the field that you specified. This is displayed by choosing option "G".

N = $

This option will select all those records with, for example, all those individuals born in the same year, or all those vehicles described as red. You must however state exactly and precisely what is in the field,

O >= $

In a numeric sort this option will select for example all those countries with a population greater than or equal to that which you specified.

P < $

If this option is used on an alphabetic field then it could for example make a sub file in which the names of people are lower in the alphabet than the particular letter you specified.

Q CHARACTER GROUP

This is perhaps the most useful of the first five options. Unlike the other four, the field contents to be searched do not have to be precisely stated. For example, it will search all the fields under NAME for "log" and come up with JOE BLOGGS and any other names that contain that combination of letters.

GENERAL NOTES

In any of the options above, you are informed if the result is POSITIVE or NEGATIVE before being returned to the SUB MENU. Once a sub file has been created, all the options in the sub menu operate on THAT sub file, not on the main file. This includes the printer option. Once option "W" is selected and you return to the main menu, you lose your sub file, unless you SAVEd it under option "R".

R - SAVE SUB FILE

After using the SEARCH, SHRINK and SORT options, operating on and reducing the main file and subsequent sub files to a final acceptable result, you can save it to disk. What you create in effect is another new main file in its own right. This should of cource be saved under a new name.

S - DISPLAY SUB FILE

If you enter this option before you use any of the SEARCH, SHRINK and SORT options, then what will be displayed is the main file. Obviously you cannot display a sub file if you have not created one. The format is the same as option "D" in the main menu.

T - SORT SUB FILE BY ANY FIELD

This operates on the sub file in the same way as option "G" in the main menu does on the main file.

U - DELETE SUB FILE FROM MAIN FILE

After using option "Q" in the sub menu, this option will leave you with a reduced main file. In effect you will have split your original main file into two main files. Selecting this option automatically returns you to the main menu.

V - LIST FIRST FIELDS

This provides a list of eight of the first (main) fields in alphabetic or numeric order is used after option "T". The U and D keys operate in the same way as option "D" in the main menu. Only eight fields are listed at once in case all the fields have used two line entries.

W - MAIN MENU ** DESTROYS SUB FILE

Once you have left the sub menu you cannot return to it to view your sub file. The only way you can see your sub file again is if you SAVEd it and then re-loaded it, or go through the whole SEARCH process again from the original main file.

X - CHECK FREE SPACE IN FILE

This may take a few moments, but it will tell you how much room you have left for more records. Assuming that all four fields in every record use up 78 characters.

Y - SEND TO PRINTER

This option first gives you a choice of two routes. You can either print out the whole file or sub file as the case may be. This may take some time and a lot of paper if you have a large file. Or you can choose to print a specified single record. If you choose the FILE option it can only of course print out that file that is present. If you are using a CBM printer you are then presented with a choice of upright expanded printing in one of six colours or one draft style in black. If you are using a STAR printer you are offered a choice of upright expanded or italic expanded printing in one of seven colours, plus one draft style in black italic.

An escape back to the sub menu is provided. If you are printing a file you are now asked to check that your printer is switched on or you can escape again if you wish. If you are printing a record you are asked which one by the first field. If you cannot remember it all, enter the first few letters or numbers. The computer will make an effor to offer you a record to try. Again the chance to escape is offered. An additional feature when printing a file using either printer is a choice of fields in addition to the main field. You can print all four fields or a choice of the second, third or fourth. This is useful if only a limited amount of information is needed.

Z - DIRECTORY DISPLAY

This is exactly the same as option Z in the main menu.

FURTHER ADVENTURES IN 'C'

We continue our look into the possibilities of making 'C' the alternative language for all C64/C128 users

JOHN SIMPSON

"The essential aim of a good programmer should be to produce sequentially flowing program code, with no unconditional jumps backwards or forwards from section to section of code".

STRUCTURE

Any code which is executed repeatedly should be so constructed that it is repeated from within a control loop, and the transfer of program control should always be carried out by using the **if...else...do...while** command statements, or the logic equivalents of these (and, or, not).

There are some programmers who will often compress several expressions into just one large expression which is then sometimes used as an impressive example of 'logicus complexicus', but really the only function it will serve is to make the program virtually unreadable. Personally I would much rather write them out as a number of single lines of code. This makes things so much easier to read and understand, especially when returning to a piece of code which I may have been working on several weeks, or months, earlier.

IF...ELSE

Probably one of the most regularly used control statements of C, and, as it happens, also one of the simplest. First we shall look at a couple of formal examples demonstrating the logic of the statements.

Formal example 1; IF (expression) statement A

In this case, if the result of the EXPRESSION is found to be true then STATEMENT A will be carried out, otherwise the next statement of program code would be executed.
Formal example 2; IF (expression) STATEMENT A:ELSE STATEMENT B

Here, we can see that if the EXPRESSION is true then command will execute STATEMENT A after which it will skip the ELSE STATEMENT B. But on the other hand, if EXPRESSION is found to be false then command will be passed, via the ELSE control statement, to STATEMENT B.

Let us now examine the two formal examples with a couple of program examples;

```
/*...lesson 5...5...else...*/
main()
{
int input;
printf("Do you enjoy reading the
CDU magazine?
n");/* line 1 */
printf("strike key <Y> for yes, or key
<N> for no.
n");/* line 2 */
input=getchar();/* line 3 */
if(input == 'Y') /* line 4 */
printf("Well that is good new for the
editor
n");/* line 5 */
else
printf("Have you tried reading the
Chinese abacus monthly,
instead?
n")/* line 6 */
}
```

The lesson starts off with the usual comment line followed by main (), the only function. (Remember, we must includa a main () function somewhere with the program). Then comes our start brace for the statements which will be incorporated with the function, and next we set up an integer type of variable, assigning it with the name INPUT. Lines 1 and 2 will simply print out our message, as we discovered in lesson 2 of last month. Line 3 introduced the C command getchar(), (more about these, (), brackets in a later lesson). Getchar is used to obtain a character from the keyboard. Once the response is received, the key value is assigned to the variable INPUT (note, the command getchar alway returns an ASCII integer value). Line 4 is the first part of the if...else command. What is actually occuring here is that variable, INPUT, is tested to assess if it is equal to the character 'Y'.

We must enclose the character being tested for within single quotes because we are comparing the integer value of INPUT with a character. The quotes simply tell the compiler to convert 'Y' into it's ASCII form.

If the value of INPUT is equal to 'Y' then the result is considered to be true and so the statement of line 5 will be executed. However, if the variable INPUT does not equal the ASCII value of 'Y' then the result is false so therefore control will drop through to line 6, and execute the statement held there. After which we have the closing brace of the function. (note also the semi-colon, ';", after each statement. As I mentioned in previous lessons this is very important.).

Before we continue any further I think a quick refresh on the operations of "=" and "==" from lesson 4 might prove useful.

(A) TEST=ACTION
(B) JOHN=PAUL
(C) SUM==RESULT
(D) SEA==GOODTOSWIMIN

In the examples (A) and (B) the variable on the left is given the value of the variable on the right - 'passing values'. In (C) and (D) we have examples of equality. One would test the variable on the left to discover if it was equal to the one on the right. If it is, then the result if true if not the result is false.

The next lesson is intended to demonstrate just how the IF...ELSE command can easily produce a range of controlled branches. The variable names as INPUT will, once again, be used to store the users key response. I shall also be introducing a new command - #INCLUDE <STDIO.H> which will force the compiler to

include various definitions within the main program. Definitions that are covered by this are the EOF (End of File) and EOL (End of Line), this command is not strictly necessary within this example program but it's introduction now will be of value later.

```
/* lesson 6....Branches in a function
tree...*/
#include<stdio.h>
main()
{
int input;
reply();
input=getchar();
if(input=='t')
top();
else if(input=='m')
middle();
else if(input=='b')
bottom();
}
reply()
{
printf(".....Select a position.....\n\n);
printf("TOP.....strike key <t>\n);
printf("MIDDLE..strike key <m>\n);
printf("BOTTOM..strike key <b>\n);
printf("\n\n\n");
top()
{
printf("Will selecting the top position
bring its just rewards?\n);
}
middle()
{
printf("Does sitting upon a fence
mean you can fall either way?\n");
}
bottom()
{
printf("Is the weight of the whole
building resting upon the bottom
layer of bricks?\n");
}
```

After getting the INPUT variable as an integer, the program now calls the function, REPLY(), and executes it. In essence it prints the first line of text, with two carriage returns, which is then followed by a further three lines of text, and then three carriage returns.

Once again, as we saw in lesson 5, the variable INPUT is assigned the ASCII value of whatever key is being depressed. This value is then tested against the three IF...ELSE statements which follow. If one of the statements is found true then control will pass to the appropriate function. For example if the user selects key 't', then control will pass to function TOP() and the appropriate message will be printed to the screen after which the program terminates. However, if a key is struck which does not correspond with either 't', 'm' or 'b' then the program terminates without further action.

Finally, and to end this months lessons, we will now take a look at the DO...WHILE command statements for passing control from one statement to another. Here then follows a formal example of a DO...WHILE situation.

Formal example; Do statement while (Expression)

Here the statement will be executed whilst the expression is not true. This is a structure for creating control loops, for example, as lesson 7 will demonstrate.

```
/* lesson 7...do while or not to do
        while...*/
main()
{
int c;
c=1
do
{
printf("Do while, da dah da do
while do\n");
c=c+1;
}
while c<=4;
}
```

The single function called main sets up an integer variable called c and assigns to it the value of 1. The next program sted will do the print statemenmt and then increment the varialbe c by one. (In fact, "C" contains a very much more sophisticated method for handling increments and decrements which I shall discuss in detail in a future lesson). Finally the statement WHILE C<=4; means that while the counter c is equal to or less than 4 then loop back to the DO part of the function and execute it once more. As soon as c reaches 5 the loop has ended and the function will terminate.

In this example the test was carried out after the increment operation. This is fine when you know that a certain course of action has been carried out for the first time, or iteration, or when you wish the loop to terminate after carrying out a statement. We can also write a few lines of code without using the DO part of the DO...WHILE combination, and that will increment c before execution of the statement. Lesson 7(a) is an example.

```
/* lesson 7(a)..not so much a do, just
a while..*/
main()
{
int c;
c'1;
while(c<=4)
}
printf("While away the hours
without the hint of a do\n");
c=c+1;
}
}
```

As I said, the test for C reaching it's optimum value has been made before execution of the statement. This allows the loop to be terminated after the increment and before the statement is executed. (NB: Note the way the braces are used to divide the different program sections in the example programs. This allows one to follow the program logic much more easily as we shall see in leater lessons which will give much more detailed and complicated programs).

SUMMARY

1. The introduction of controlled program flow using the IF... THEN...DO...WHILE expressions.
2. The introduction of the "C" command GETCHAR(); which is used to pick up character input from the keyboard.
3. GETCHAR(); always returns an ASCII integer value.
4. The introduction of #INCLUDE<STDIO.H>
5. Several functions were included in lesson 5. Note that at the completion of a function the program terminates.

Later I will demonstrate the three methods of exit from a control loop or function, these are namely, BREAK...CONTINUE...RETURN.

LETTER MAKER

Another novel way of sending information to friends or writing program notes is catered for

ROBERT TROUGHTON

Letter Maker is one of those "fun" utility programs that pops up from time to time. It does not really provide a utility in the true sense of the word, but is fun to use and can produce some fairly impressive results.

The idea behind the utility is simple. It allows the user to produce nice pages of text that are both colourful and musical, without the normal laborious job of writing long lenghty letters. (lets face it, who really enjoys writing letters to friends and reletives anyway??).

Using the program is simplicity itself. Just select the option from the main CDU menu, once activated sit back and compose your screens. Below is a quick run down of available options. There are plenty of on-screen instructions so you should not have too much difficulty in using this simple, short but fun utility.

MAIN EDITOR

F1:	Page forward
F2:	Page backward
F3:	Centre line
F5:	Options menu
DEL:	Delete character
CLR:	Clear screen / home cursor
INST:	Insert character
CRSR:	Move cursor around screen
HOME:	Homes the cursor
CBM I:	Insert a line
CBM D:	Delete a line
RETURN:	Carriage return

Thanks must go to Richard Rinn (DEEK) for providing me with the music that accompanies this program. Thanks DEEK.

C64 AMIGA C128

ONLY £149

C64 DISK DRIVE

FREE GEOS

COMMODORE 1541 C11 DISK DRIVE, SLIMLINE CASE, POWER SUPPLY UNIT, 5¼" DRIVE £3.50 + P&P
THE ONLY DRIVE 100% COMPATIBLE WITH THE C64/128

NEW

AMIGA 500

ONLY £379

FLIGHT OF FANTASY
★ F29 RETALIATOR ★ RAINBOW ISLAND
★ ESCAPE FROM THE PLANET OF THE ROBOT MONSTERS
★ DELUXE PAINT II
£5.00 P&P

NEW

LIGHT GUN £36.50

C64 LIGHT GUN PLUS GAMES AND UTILITIES
LIGHT FANTASTIC UPGRADE PACK
£3.50 P&P

PRINTERS

COMMODORE MPS1230	£159.00
COMMODORE MPS COLOUR	£219
SEIKOSHA SP .180V (C64)	£149.00

C64 LIGHT FANTASTIC

LIGHT FANTASTIC PACK INCLUDES - LIGHT GUN 3D GAMES, PAINT PACKAGE, DATA RECORDER, GAMES
£140.00 + £5.00 P&P

100% ERROR FREE DISKS

10x3.5" DSDD	£9.50
10x5.25" DSDD	£4.50
50 DISK BOX 5.25"	£5.99
100 DISK BOX 5.25"	£6.99
80 DISK BOX 3.5"	£6.95

C64 DATA RECORDER	£24.50
'LOAD-IT' DATA REC	£35.00
C64 POWER SUPPLY	£19.95
C64 MOUSE, MOUSE MAT & HOLDER	£26.50
KONIX JOYSTICK	£11.50
MOUSE HOLDER	£2.50
MOUSE MAT	£4.50

CARRIAGE £1.50

C.M.S CROFTON MICRO SUPPLIES
45 WHITBREAD ROAD
BROCKLEY LONDON SE4 2BD
081-469 3246

COMPUTERS

Amiga 500 FLIGHT of FANTASY PACK comprising: TV Modulator/F29 Retaliator/Rainbow Island/ Escape from Robots/DPaint II	359.00
Amiga 500 BATMAN PACK: Batman/F18/NZ Story/DPaint II	359.00
Amiga 500 CLASS OF THE 90's Education Pack	529.00
Amiga 500 + 1084S Stereo Col Mon	605.00
Commodore 64C LIGHT FANTASTIC Pack comprising: 64C + C2N Light Gun + 3D Glasses + 6 Games + Paint Prog + Music Prog + Typing Tutor + Arcade Construction Kit + Audio Tape Align	139.99
Commodore 64C WORLD CUP Pack comprising: 64C + C2N, 2 Joysticks, 5 Games	149.95
Commodore PC Starter Packs (AS SEEN ON TV)	FROM 608.35

PRINTERS

Citizen 120D Parallel or Commodore	129.95
Star LC-10 Parallel	159.00
Star LC-10 Colour Parallel	205.00
Star LC24-10 24 pin Multi-font 170/57cps	239.00
All Okimate 20 consumables normally in stock	PHONE

MONITORS

Commodore 1084S Stereo Colour Mon	259.00
Philips 8833 Stereo Colour Monitor	249.00
Philips 7502 Green Screen Monitor	95.00

MICELLANEOUS

Amiga 512K RAM/Clock Exp for A500	65.95
Commodore 1541-II Disk Drive	129.00
Commodore C2N Data Recorder	29.95
External 3.5" Disk Drive for Amiga	69.95
Power Supply for C64	26.45
Super-G Parallel Interface for C64/128	34.95
Star NL-10 Interface for C64/128	39.00
Surge Protector 13A Plug	12.95
Surge Protector 3-Way Adaptor / 4-Way Dist Unit	19.95/15.95

SOFTWARE

VizaWrite 128 + Spellcheck	49.95
VizaStar 128 Spreadsheet & Database	59.95
Superbase 64 or 128	29.95
Superscript 64 or 128	32.95
Tasword 64 40/80 Col WP - Tape or Disk	24.95

ALL PRICES ARE INCLUSIVE OF VAT AT 15%
CARRIAGE £5 (EXPRESS £10).
Prices subject to change without notice E&OE

Delta Pi Software Ltd

8 Ruswarp Lane, WHITBY, N. Yorks. YO2 1ND.
Tel: 0947 600065 (9am-7pm)

ACCESS VISA

FUNCTIONS

If you have ever wanted to make better use of those virtually inaccessible function keys, then this program is the one for you.

STUART WESTBROOK

With this utility you can assign strings and character codes to a total of sixteen function keys, and save a machine code file, maximum 200 characters, which can be loaded on top of a BASIC program without destroying it. The function keys are defined for use in DIRECT MODE only, and their aim is to be an aid in the development of BASIC programs. The sixteen keys available are accessed as follows:

FKEY NO. ACCESSED VIA
1-4 F1, F3, F5 and F7 ONLY
5-8 F1, F3, F5 and F7 PLUS SHIFT
9-12 F1, F3, F5 and F7 PLUS COMMODORE KEY
13-16 F1, F3, F5 and F7 PLUS CTRL KEY

I will now describe each of the available options in turn.

VARIOUS FUNCTIONS 1,2 and 3

These are the first 3 options available from the main menu, each offering eight of the most popular {fixed} definations. Details are given concerning the method in which your next defination will be accessed, and on selecting a defination you will be offered a carriage return.

STRING FUNCTIONS

As various functions, except that the available definations are all under the category of String Functions.

DIRECT DEFINATION

Where as the previous options consisted of FIXED definations, this option gives you much more versatility as functions can be defined more specifically. Selecting options 1-7 will allow you to add to the defination chosen, for example;

```
LOAD'$',8
POKE53280,x
SYS64738
```

Obviously, to defined every possible command/statement would be impractical, so selecting option 8 allows you to define anything you like, simply enter your defination {carefully as you cannot use the delete key} and press return.

EDIT OPTIONS

After selecting edit options you will be presented with the following options;

1. INSERT CHARACTER CODE (CHR$)
Select the function number that you wish to define as a character code, and enter any code between 0-255. Refer to any manual for a list of codes.

2. DELETE DEFINATION(S)
You are given the opportunity to delete ALL definations, replying 'N' will give you the option to delete any SINGLE defination.

3. REPLACE DEFINATION
After selecting the function number to be replaced, you are returned to the main menu to redefine the function chosen.

4. SWAP POSITIONS
Enter the first function number you wish to swap (press return), and then enter the second. If either of the function numbers equal 17 then you will be returned to the main menu with no alterations made, otherwise the definations are swapped and you are returned to the edit options menu.

PRINT DEFINATIONS

Remembering how you have defined each function can be very difficult, and for this reason there is the option to print a list containing each function number, it's access method and defination. You are given a prompt concerning whether your printer, device 4, is ready for use or not. To continue when the printer is not ready could mean the loss of all data or may even crash the program. The printout may contain blank lines, this indicates that the defination above the blank line contains a carriage return. Additional details are added to the end of the list.

SAVE MACHINE CODE PROGRAM

This is the programs most important option, it allows you to save your definations in machine code format to either Tape or Disk, meaning that it will not only load much faster but it can also be loaded over a Basic program without destroying it. Again, be sure that the device chosen is ready for use, otherwise it could result in loss of data or worse. Please note that the machine code is loaded independently of the FUNCTIONS program.

QUIT PROGRAM

If you have completed your definations and saved your machine code version, you may quit the program and execute the machine code by SYS49152.

THE MACHINE CODE

Once saved, this is loaded independently of the FUNCTIONS program and is loaded as follows; LOAD'filename',x,1 where x is the device number. To execute the machine code enter SYS49152. To disable the function keys type PRINTUSR(1) to reactivate enter PRINTUSR(0).

ADDITIONAL KEYS

CTRL and A equals Auto repeat
CTRL and C equals Remove auto repeat
CTRL and @ equals Pause system
CTRL and ^ equals Remove pause

DEMONSTRATIONS

To demonstrate the use of FUNCTIONS there are 2 example files on the disk. To see them in operation, select either DEMO 1 or DEMO 2 from the menu, or load direct. Once in memory you may activate them by SYS49152.

Adventure Helpline

Jason Finch **provides more help for those that are stuck with CDU's Kron adventure.**

Here we are already with the third article in this section of the Adventure Helpline series, covering the epic adventure Kron, written by Tony Rome. In the two previous articles I have explained, through subtle (or otherwise!) clues, how you can make your way through the first two stages of the adventure - the Sea of Storms and the great subterranean system of caves. Last month I left you in the Valley of the Dead looking at the Castle of Spells that is over a deep lake to the south. The problem was how to get over this lake and into the castle. I provided you with two cryptic hints and hope that you have successfully made it across. But in case you are still hurling large objects at your computer, the next paragraph explains how it is done.

Firstly, do you remember when you released an eagle when it had become trapped in a crevice some while ago? It left behind a flute and you will find that the eagle will return to you in the Valley of the Dead if you play this flute. If all goes according to plan, the eagle will swoop, pick you up and carry you over the lake, letting you drop safely onto the castle battlements. You are now in the third and final stage of the adventure.

This section contains possibly the most important and baffling sequence of puzzles - how to get passed the prison guard and rescue princess Zora without being killed by the ogre or falling asleep! That is not an insult to the author - you will find out what I really mean a bit later on. Once you are on the battlements, the only way out that you can see is by a staircase that leads downwards. This would seem the most obvious way to enter the main castle and so do that - go down. You should find yourself at the foot of the stone staircase with a passage to the north. You should follow this passage until you come to the Alchemists Chamber. Take whatever is on offer but do not open it - the jar contains a strange potion that induces tiredness. Also, make sure that you do not stay around too long in the laboratory.

At this stage you should walk straight through the large hall that is found on the other side of the staircase. You will then find yourself in a magic room with the walls slowly closing in around you. I shall be a little cruel here and refuse to tell you what to do! - just remember what the guru told you was on the scroll because this is the time that the walls may move your senses! One step in the wrong direction and you will be confronted by an ogre that kills you immediately.

When you have found your way into the castle dungeons you will meet the guard who is keeping watch of the princess. Here you will need the jar of sleeping potion. Open it and the drop it on the floor. It takes a little while to act but find somewhere else to wait or else you will find that it is you that is asleep. It is best to back-track your last two movements, north to the circular room and then whichever way it was to get back to the passage that originally lead you into the circular room. Then come back again straight away to the dungeon. That will take four moves by which time the guard should be in the land of nod.

You can then search him and rescue the princess. But your quest is not finished because you must still get back to the Cave of Ice. And of course you must negotiate that circular room and set of passage-ways correctly again to avoid meeting a very unfriendly ogre! You should eventually end up back in the large hall with princess Zora. However, Balzan is waiting for you and fires a deadly beam right in your direction. You should find a way to shield (clue) yourself and reflect his beam straight back so he hits himself.

You will then be able to open the chest and grab hold of your findings. You may also find that a candle will be of use when you go south. Once you are in the room you should look around once again to recognise where you are. Examine one of the mirrors and then do what they do in all the good films to what you find. Look around once again and then with the candle make your way down the staircase that you have just uncovered. If you do not take the candle with you then it is goodbye adventurer!

You will find yourself trapped, confronted by a silver door to the north. Now what else have you found back in stage two that was silver?? Upon examination of the door you are told that only those who know the password can open it. So just type in whatever was written on the particular item from stage two and hey presto the door should swing open so make you escape. Now you will finally find yourself miraculously transported back to the Cave of Ice which you were told holds the key to completion of the game. Ice and snow cover the roof but just try examining the roof a bit further and something may just uncover itself! I will not tell you what to do here. It is sufficient to say that there is a cord and you have a knife that has not been used yet. Recover the crown and the give your magic ring a rub. You will be whisked away and now that you have Zora with you, the adventure has been successful and is complete. Well done!!

Next month I shall provide you with some more invaluable information regarding Kron and its many locations and puzzles - some more difficult to solve than others as this article has hopefully shown! This will allow you to complete Kron very easily and relatively quickly so until then - have fun adventuring!!

GAMES LIST creator

Keep a record of all your games disks with this versatile and novel utility

JOHN KAY

Games List Creator is a utility that enables you to keep a record of all your games disks, which when run will display your lists in a pleasing and musical manner. To use the program is simplicity itself as no knowledge of machine code programming is neccessary. First of all take a blank formatted disk and copy the program GAMELIST CREATOR from the CDU disk. Load and run the GAMELIST CREATOR program which will then present you with the main menu screen. There are four options on the menu which are;

1. Create a new list
2. Add to an old list
3. Create a new scrolling message
4. Run games list

Option 1 is the first one that you will have to use. {Options 2 and 4 will not work unless you have already used option 1}.

First of all to create a new list take option 1. When activated you will be displayed a message telling you to type "Q" when finished. Remember this. Press any key to continue with your choice. Displayed in the top left corner is an asterisk, this is now your cursor. Below is a message asking you to enter the title of a game, then a message saying how many games you have entered and finally a message informing you of the option you are in. To enter a game title simply type the name and press return to store the name in memory. Repeat this operation until all your titles are entered. Once finished, type "Q" and you are then ready to save your list to disk for future recall. First of all, any old "Games Lists" are wiped and your new list is saved to disk.

Consider you have now just purchased a few more new games and you want to add them to your list. Simply load and run your newest Games List program and select option 2 from the main menu. Entering the new names is the same as for option 1. Please note that whilst using options 1 and 2 any characters except # and $ can be used. A game title can be up to 29 characters long. The program allows for 14K for the list, which should be sufficient for over 1000 titles.

CREATE NEW SCROLLING MESSAGE

In the bottom border the program incorporates a scrolling message. What this message is, is entirely up to you. When you take this option, you will first be asked the question Size of space (1-5)? This means how much space you want between each word on the message. It is recommended that 2 or 3 is chosen. You will now see the asterisk at the top left of the screen and the message "End here", at the bottom left. Type in your message using the keys A-Z only. Do not press the return key until you have finished typing in your message. Do not worry about the words splitting or appearing to be joined up. The program incorporates a word wrap facility. Once the program has read your message and stored it on disk, press any key to continue.

RUN GAMES LIST

Now for the big one, your list is up to date, you have created a nice witty message and you now want to see it all in action. Option 4 of the main menu will enable you to do this. The program will display a list of 2 files which will needed to be loaded, press any key to start the load. When they have finished loading any key will run the games list. When run you will be faced with a static screen, this is for people with cartridges that wish to freeze out the program. The static screen drops a little hint that it is waiting for you to press fire on your joystick. So insert your joystick into Port 2 and press fire. Once you have viewed a screen press fire for the next screen. Have fun!!

Typing "EXIT" will return you to the main menu.

DUAL DISKCOPY

At last, an Intelligent Twin Single drive disk copier is provided for those luckily enough to have more than one disk drive

MIKE HOLMES

Oh no! not another disk coypying program (yawn). Let it be said at the outset that this particular disk copier does not pretend to be so marvellous as to make all other disk copying utilities obsolete overnight, but I hope you will find it different enough to be equally useful where the occasion demands.

Disk copying utilities come in many forms. Some may be provided as part of a word processor or other software package for making backups, or they can be short Basic programs with a little machine code support as can be found for instance in the book "Anatomy of the 1541 Disk Drive", or they can be 'turbo' backup cartridges, or provided as part of an alternative operating system such as GEOS. They each have their own merits. Geos can copy individual files or a whole disk very fast; the simple Basic program versions are convenient and simple to use, if somewhat slower.

There are a few anomalies with the majority of copiers in that invariably, for example, the copy ends up as a complete exact relica of the original, because the copy has been made exactly the same, that is to say sector for sector. This means that, due to much re-saving, modifying and extendinf of your files, some of these have consequently become 'spread about' the disk because there are no more 'local' sectors free. Because of this fact, your copy will be in the same mess as your original. It would be nice if one could make a tidied-up version instead, each file saved to the copy individually, and not as part of a mish-mash pattern of blocks. Perhaps even the directory entries can be re-organised in a more orderly manner.

Secondly, there is the irksome business of disk swapping, though if you have only one drive you have no choice. But if you have two, some copiers still insist on the disks being swapped as they only have the other option of copying to the back of the same disk with a double sided drive (eg. Superscript). Geos will copy from device 8 to 9 and vice versa but not without the first problem mentioned above. And of course, many copiers format the destination disk as new as a matter of course, in order to be able to put down this duplicate image pattern, as naturally it is not expected that there may be any existing files on the destination disk to be preserved. If there are, these will disappear. But then some copiers cannot format the disk themselves, just to add to the confusion. So what if you just want to add the contents of disk A to those already existing on formatted disk B? Copy each file individually? A process that can become very tedious to say the least.

Suffice to say then that to put an end to all these inconveniences, "Intelligent Twin Single Disk Drive Copier" was created. It is 'Intelligent' in that the program tries to cope with any probable complications that might arise in the copying process, like for example, some source file won't read properly, or the destination won't write or verify due to either corrupt data or misaligned tracks. Such problems usually completely crash a conventional copier, and of course, if the destination copy hasn't been properly finished off, you can't even access the information that has been saved onto it so far. Also, as a further aid to the poor human operator, Diskcopier does not require endless disk swapping, as two seperate drives are used. Therefore, having got Diskcopier on its way you can then leave it to get on with it. Another variation on a theme, then, which I'm sure you will find a useful addition to your armoury of uitlity programs.

To start, BOTH disk drives must be ON, and one of these must ALREADY be configured as device 9. The program is loaded from device 8 by LOAD"DISKCOPY.BAS",8,1 or select "DISKCOPY" from the main CDU menu. The screen fills up with rubbish from the top down and then a little while later the start-up message will appear. Below this is the message "Drive a: = device 8 - drive b: = device 9 - Insert Source Disk in Drive a: and Destination Disk in Drive b: Key RETURN when ready". This being done, keying RETURN brings you to the selection from a directoy list stage.

Diskcopier does not copy sector by sector, but file by file using a list of filenames from the directory of the source disk. The message is "Enter directory mask '$' followed by wildcards '*?' and/or filetype identifiers 'p/s/u' as necessary." Below this the first required symbol, '$', is

already given as part of a line to be INPUT from the keyboard, followed by a flashing cursor. You can press RETURN and Diskcopier will assume that you want everythin, (ie. '$') to be copied from the source disk in the order that it appears in the directory. Alternatively, you can re-organise the file entries for the destination disk using the wildcards or filetype identifiers. '$temp*' for example, will only produce a list of files to copy beginning "temp", of any filetype. Or, '$temp*=p' will only produce the same but which are PRG (programs). Or you could copy all 'programs' first with '$*=p', then when Diskcopier has finished with these, rerun it to copy all the SEQ files with '$*=s', and then again for anything else, which can be done with just '$' again.

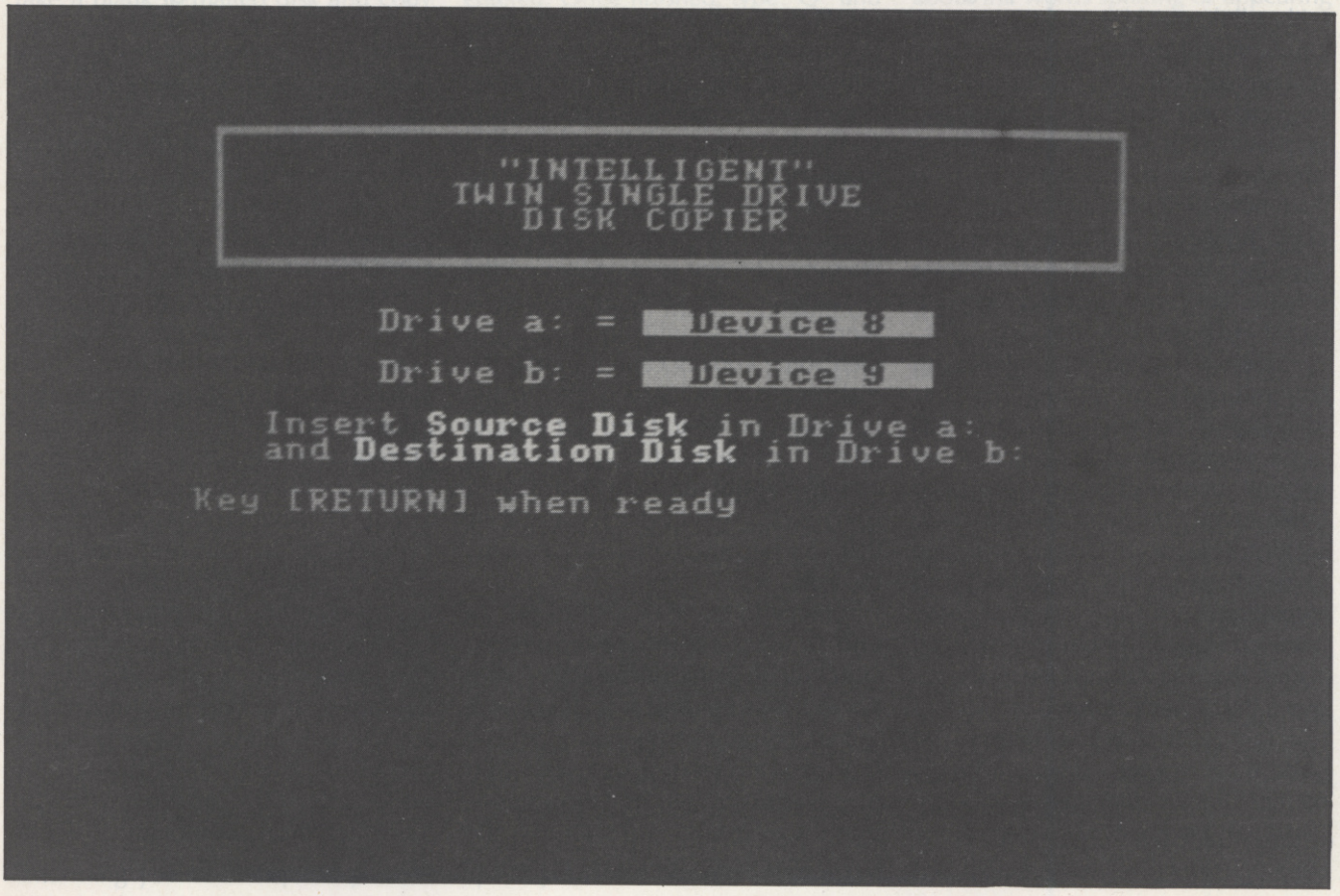
This last is possible because, after you've written a directory mask, Diskcopier loads in the directory from drive a: according to the mask. This is also listed on the screen so that you can see what you're going to get, in the usual directory format. If you didn't quite get what you wanted it is possible to abort Diskcopier with RUN/STOP RESTORE (but do it quickly), and to restart it with 'r'.

After listing the directory, another message appears, "Overwrite any duplicate files (y/n)?", and you just key 'y' or 'n' as appropriate. It is this feature that makes it possible to copy a select group of files first, and then anything else otherwise left out by universal mask '$', because keying 'n' to preserve files with duplicate names causes Diskcopier to ignore the files it's already put on the destination disk, and instead go directly to the next. It also provides the means of protecting data on the destination disk if it already has files to which those from drive a: are being added, just in case there are any with identical names. The default is 'n', so even just keying RETURN for this last query will ensure the safe option of no over-writing.

From this point on, Diskcopier requires no more human intervention, so after a hard day's programming or word processing you can leave Diskcopier to make your backup copy while you go away and have your tea. (It might be prudent though to hang around a minute to ensure that the next stage is completed without any hitches).

Because the very next thing that Diskcopier will do is to initialise the destination disk (drive b), which causes the disks ID and BAM to be read by the DOS. If this fails then either there is no disk in the drive, in which case Diskcopier is going to get nowhere, or the disk is a new one straight out of the box and so is unformated. If a disk is unformatted then Diskcopier formats the disk as new, using for convenience the diskname and ID of the source disk, automatically. Furthermore, and even if the disk is formatted, the destination disk is tested.

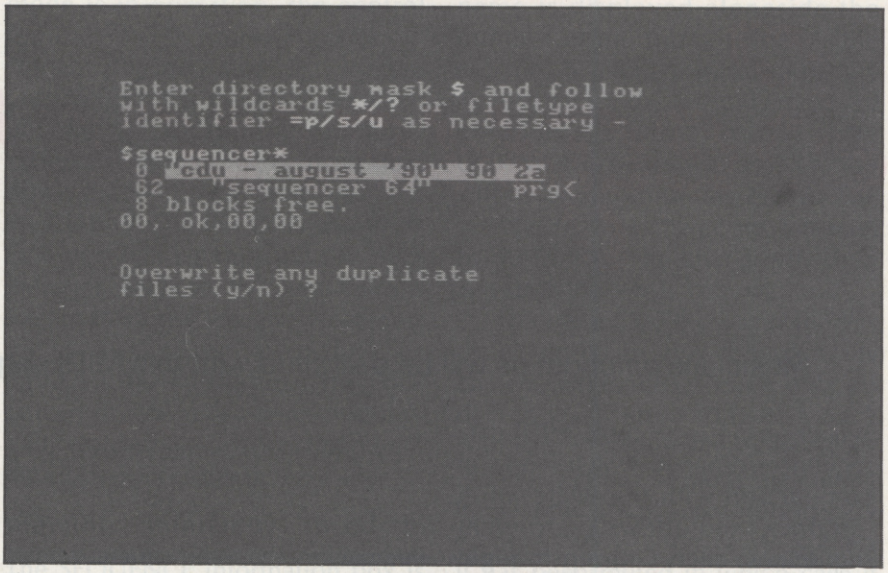
If the destination disk initialised ok, then Diskcopier assumes that at least the directory track can be read from and written to alright, and hence also the tracks adjacent. But in addition, and even if a new disk was formatted, a check is made on the two opposite extremes of the available disk surface. ie: tracks 1 and 35. All sectors of track 1 are test loaded into a DOS buffer, and then similarly all sectors of track 35. If there were no


"INTELLIGENT"
TWIN SINGLE DRIVE
DISK COPIER
Drive a: = Device 8
Drive b: = Device 9
Insert Source Disk in Drive a:
and Destination Disk in Drive b:
Key [RETURN] when ready

problems then it would be fair to say that the disk if a healthy disk in drive b:. If any of these tests fail then, instead of reformatting the destination disk to make good - a definate no-no, since it may have valuable data on it already, even if it is a bit out of alignment or old and worn - Diskcopier throws up this quaint fatal error message in lurid pink; "Serious problem with destination disk. Operations aborted, unsafe to continue. Replace or reformat disk in drive b: before re-attempting copy"

If however everything's gone according to plan, then the files will be copied one at a time, in sequence using the directory list. At the commencement of each file a line appears on screen. "Copying a:<filename>", and below this a line of blue dots should appear. The appearance of each dot signals the successful transfer of one equivalent sector full of data. I said that Diskcopier does not copy sector by sector, but for convenience it still transfers data in 254 byte chunks, since one character at a time is much too slow. the very last chunk may not make up a complete 254 byte, so just those remaining to the end are sent. This display of blue dots is included as a diagnostic aid; if the display freezes then a drive has hung-up and has to be reset. Each dot should appear after an interval of around one second - any faster means that either drive a: is not reading or drive b: is not writing, in which case Diskcopier must be stopped with RUN/STOP RESTORE and restarted from the beginning. If this still doesn't work then the drives must be reset and re-configured, and the whole process restarted from scratch.

A screen display builds up of the filenames copied with their adjacent sector count (in groups of blue dots), which is useful for verifying the file sizes. One or two problems can occur here. If some part of the source file won't read properly, Diskcopier abors further attempts and announces, "Can't read a:<filename>, going to next item". However, everything that has been successfully lifted so far IS saved to the destination, and the



destination file is closed properly. You then at least have a properly structured partial copy.

Whether or not the previous file copied completely, the next filename is extracted from the list and this copied next, and so on. The other serious error is where it is discovered that the destination disk is not that good after all (both drives are continually quizzed for errors), in which case we get the pink "serious problem" message again, and the program stops. BUT, whatever state the destination disk is in it has something on it that is retrievable by normal means.

The last problem that can occur is that the destination disk become full up, because the files that have just been transferred to it may not all be all the it has in total. (Possibly had some files on it already). This generates the "Disk full" message, and then Diskcopier erases the file it just tried to save but which didn't fit, and ends.

Diskcopier comes into its own when making a good copy from a lousy "dodgy disk", with misaligned tracks, corrupted sectors, scratches or whatever. Diskcopier won't be put off until it has at least attempted to read and transfer something from every single file, however long it takes. The ratty source disk may put up a vailant resistance, but Diskcopier won't give in without a considerable struggle. You cannot quarantee that such copies will be perfect, but at least you may have something that can be rescued.

At the very end, Diskcopier generates a report list of any files that proved impossible to read from source completely. The list caters for up to a maximum of ten filenames with read problems, which I would have thought was quite enough. This is so that if you've been away, you will know that one or more copies are incomplete on the destination disk. Any eleventh and above problem files are ignored since the list is already full. But if there were no read problems you will never see the report message "Incompletely copied:-" and the list.

Generally the time taken to copy a disk is proportional to the amount of data transferred. A completely full source disk should take around twenty minutes, and correspondingly less time for less storage or a specifically selected directory list. The standard DOS read/write techniques are retaind, if a bit slow, for reliability. For the reason Diskcopier is independent as far as is practicable, so that while it is copying a disk you can be getting on with something else.

And finally, you can update backups of 'working' disks through the 'y' option of the "Overwrite any duplicate files" query prompt. Diskcopier always erases duplicate files on the destination first, then saves the new drive a: version as new, as opposed to using the less than ideal "@:" command.

Sequencer 64

Musicians amongst us will be delighted with this MIDI sequencer for the humble C64

STEVE CARRIE

Sequencer 64 is a simple MIDI data sequencer program which works in conjunction with the 'Datel Midi Interface' and allows up to eight sequences, each with up to eight tracks, to be recorded and then played back either singly (sequence mode), or in a user-set sequence (song mode). In song mode, it is possible to play back up to 8 sequences, each of which may be repeated several times.

Track edit facilities are provided to allow simple or extensive corrections to be carried out on recorded data. A track-timing auto-correction facility is also provided.

Screen diplays are easy to follow and certain features available through the function keys are displayed along the bottom of the screen. The status of any track in the currently selected sequence is displayed through a set of track-status indicators.

It is possible to save single sequences or complete sequence-groups (songs) to disk. Disk catalogue and command facilities are also available. Sequences, and each of their tracks, may be given names and a MIDI channel re-direction facility is included.

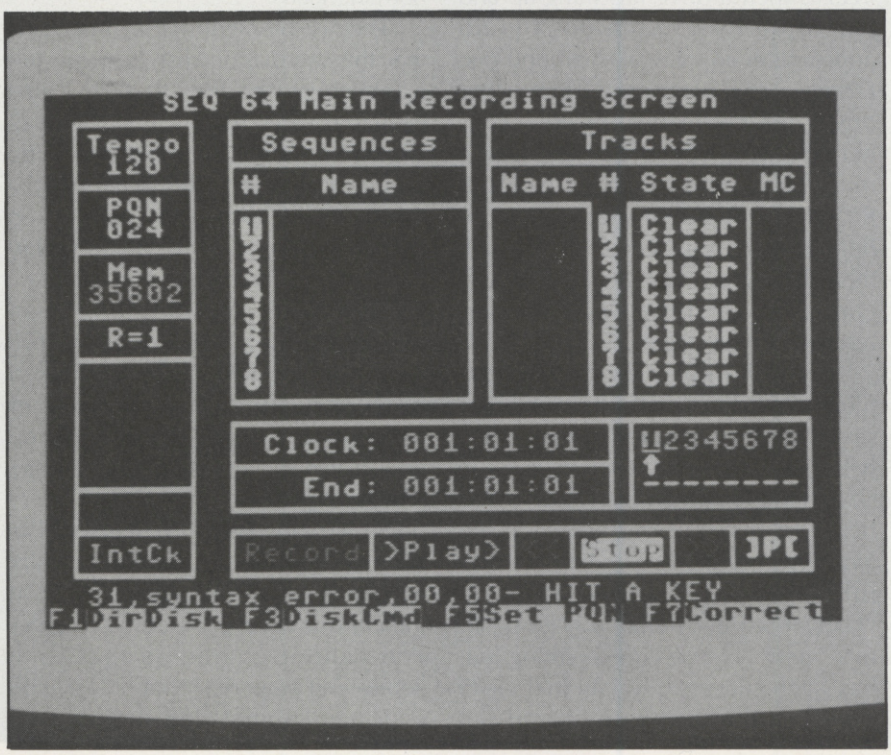
The program generates an internal clock with recording resolutions of 24 or 48 pulses-per-quarter-note (PPQN). MIDI timing data may be sent out through the MIDI OUT terminal along with normal channel data. The internal clock may be by-passed and timing information received through the MIDI IN terminal thus allowing the program to be synchronised to external equipment.

MAIN SCREEN LAYOUT

The screen is arranged into a number of boxes containing different information about the current state of the program. These are the General Info, Sequence Info, Track Info, Clock/Song Info and Control boxes. The bottom line on the screen displays 4 possible operations accessible via the four function keys. A further 8 options are available when the SHIFT or COMMODORE keys are held down (four options each). On pressing either COMMODORE or SHIFT keys you will see the bottom line change to display new options. Above this, there is a blank line which is used at various times to display information or request further keyboard input. This is the STATUS/REQUEST line.

CLOCK/SONG INDICATOR BOX

In the middle of the screen (well, almost) you will see the CLOCK/SONG indicator box. Rather than counting time, this clock counts musical intervals based on a 4/4 time signature (4 beats to the bar, beat on quarter note). The rightmost pair of digits count the number of PPQN, the middle pair count the number of beats (1/4 notes) and the leftmost three count the number of measures (bars). There would appear to be two clocks in this box, but one is the sequence end indicator which simply shows at which measure and beat the current sequence ends on (in other words, the length of the sequence). As the clock counts up, a metronome click will



sound through the TV/Monitor speaker on each beat. The click on the first beat in the measure will sound higher than the rest.

This box also shows information relating to the organisation of sequences in a chain (song). The numbers along the top and the arrow below show the current position in the song (the SEQ song pointer - not to be confused with the MIDI song pointer). This may be moved by pressing the up-arrow key. In order to make up a song, you must first assign one or more sequences to song positions. You do this by holding down the CTRL key and pressing one of keys 1 thru 8. A number will appear in the space below the song pointer. This is the sequence number for that song position. Keep pressing CTRL <number> until the desired sequence number is shown. If a dash is shown, no sequence has been assigned. You may assign the sequence numbers in any order, as many times as you like up to the limit of 8 sequences, and since each sequence may be repeated up to 8 times, the potential for fairly long musical sequences exists, limited only by the memory of the machine.

In order to play the song, you must put the program into SONG mode by pressing the M key. You cannot record MIDI data in song mode and therefore the RECORD function is disabled. The song is started by pressing the PLAY key (Space Bar) and will continue until the SPACE BAR is pressed again. Should the External Clock Source option be selected, pressing the STOP key on the external source will halt the song.

CONTROL BOX

Above the status line, you will see the control indicator box. The controls here are analogous to a tape recorder with PLAY, RECORD, FAST-FORWARD, REWIND, STOP and PAUSE. The SPACE BAR alone functions as the PLAY key, but when pressed along with the SHIFT key (effectively the RECORD key), the program will begin recording. In RECORD or PLAY mode, the SPACE BAR acts as the STOP key. Whatever mode is in operation, the control indicators will reflect the current status. If you have just loaded SEQUENCER 64, the STOP indicator will be 'lit'. Assuming you have connected up your equipment, hold down the SHIFT key and press the SPACE BAR. The RECORD and PLAY indicators will light up and the clock indicator will begin to count. If this is the first track in a sequence, recording will continue until the STOP key (SPACE BAR) is pressed. The clock end indicator will be set and any further recordings will continue until the clock reaches this point.

SEQUENCE INFORMATION

The SEQUENCE information box is located in the top-centre area and is composed of two columns. The first of these is the sequence number column containing eight numbers corresponding to eight sequences. The CURRENT SEQUENCE is shown by the highlighted number in this column. The current sequence is chosen using the SHIFTED < (previous sequence) and > (next sequence) keys. The TRACK information box will be updated to reflect the status of the tracks in the selected sequence. Alongside each number, there is a nine-character space for the sequence name. You may enter a sequence name by holding down the COMMODORE key and pressing F3. You will be prompted on the request line to enter up to nine characters followed by RETURN. The name will then appear alongside the current sequence number.

TRACK INFORMATION

The TRACK information box is located towards the top-right of the screen. There are four columns labled NAME, £ (track number), STATE and MC. In the track number column, one of the numbers will be highlighted. This indicates the CURRENT TRACK; i.e. the track in which data will be recorded, should record mode be activated. You may change the current track using the unshifted > (next track) and < (previous track) keys.

Each track may be given a name of up to 4 characters in length. The name is displayed alongside the number of the track in the NAME column. To name a track, move the current track indicator to the required position, hold down the COMMODORE key and press F1. You will be prompted on the request line to enter up to 4 characters, followed by RETURN. The string will then appear in the NAME column.

The track STATE column reflects the current status of each track in the current sequence. A track with no data in it will show CLEAR. A track with data in it normally shows ST'BY (standby). In record mode the current track shows *REC* (record). In play mode, a track will show PLAY> unless it has been muted in which case it will show MUTED. This system allows you to see what any track in the sequence is doing at any time.

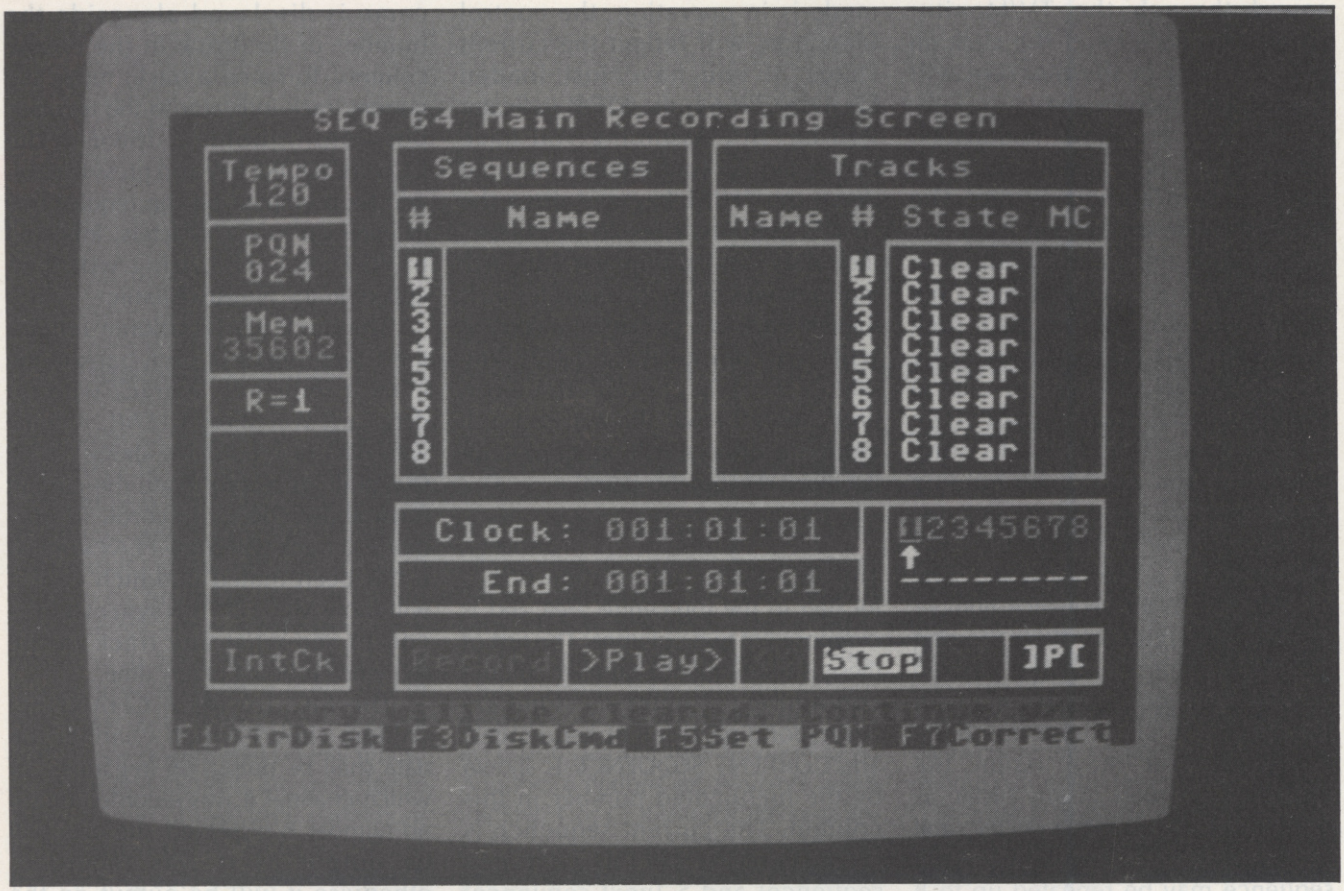
The MC, (or MIDI CHANNEL) column shows which channel the data shall be output on (regardless of which channel it was recorded on). If no number is displayed, the data will be output as it was recorded, otherwise it will be output on the indicated channel. You may change the channel by holding down the SHIFT key and pressing the key (1-8) relating to the track number.

GENERAL INFORMATION

The GENERAL information box is located down the left-hand side of the screen and is further divided into a number of areas. At the top, the TEMPO indicator displays the current internal clock speed in Beats-Per-Minute (based on a 4/4 time signature). Initially, this is set to 120 beats-per-minute. This may be altered using the + and - keys to increase and decrease this number respectively. The range is 20 to 200 beats-per-minute.

Below this, we have the PPQN indicator, intitially set to 24. You may change this using key F5 (Set PPQN). PPQN may be set to 24 or 48. The higher this number the greater the recording resolution. After pressing F5, the status line shows a number of options for the PPQN value; F1 selects 24, F3 selects 48 and F7 exits without changing the value. On

ON THE DISK



choosing F1 or F3 you are warned that changing the PPQN value clears memory and you must press Y to change or N to exit without changing the value. When using the External Clock Source option, use a PPQN value corresponding to the clock rate or your source.

Next, we have the Memory Indicator which reflects the remaining amount of free memory. When this falls below 50 bytes, recording is not possible and some data must be cleared before recording may proceed. Below the memory indicator, we have the sequence repeat indicator. This shows how many times the current sequence will play during playback. This is a value from 1 to 9 or * (infinate). You may alter this value for the current sequence using the * key. Below this we have the function indicator area. The corresponding "light" will be 'lit' when a function is activated. The various functions are described below.

XTHRU, or record playthrough is toggled using the X key. When in record mode, data received will be passed to the MIDI out terminal as it is received. Note that the enforced midi channel for the current track will be used for xthru data. (See the TRACK INFO box description of the MC controls).

SYNC, or MIDI SYNC is toggled using the S key. When activated, START, STOP and MIDI CLOCK data is sent out over the MIDI OUT terminal at a rate of 24 clocks-per-quarter-note. You cannot activate SYNC when using an external clock source.

STEP mode is activated (and deactivated) using the O key. When activated, the status line requests key input for the step interval which may be 1/4, 1/8, 1/16 or 1/32 note. (keys F1, F3, F5 and F7). When in step mode, the clock will stop after the chosen interval and you must press RETURN to allow it to advance. Press O again to turn step mode off.

CAPTV, or Captive Play Through, when activated (using the P key) allows data in the MIDI IN terminal to be passed directly through to the MIDI OUT termianl. No other function may be performed. (hence the name CAPTIVE) and you must press a key to turn it off. You can use this to test a MIDI network.

COUNT, or Count In mode is toggled using the I key. When activated, a record session will be preceeded by a two-measure count in before recording actually begins. Any data received during this time is still recorded and thus program control changes may be sent to SEQUENCER 64 before play actually begins. You will hear the metronome click during count-in through the TV/Monitor speaker.

SONG (song mode) is toggled using the M key. When active, song mode allows up to eight sequences to be played in a predefined order which is set up in the CLOCK/SONG box described earlier. In song mode, the RECORD function is disabled and you must switch song mode off to record more data.

INTCK/EXTCK. This indicates the

current clock source. The internal clock is probably accurate enough for most applications but there may be a need to sync the computer with an outside source. Toggle the clock mode using the C key. In INTCK mode, all data received is recorded, including MIDI clock and STOP/START DATA. In EXTCK mode, MIDI clock, START and STOP data is NOT recorded and is used to synchronise the computer with the rest of the system.

THE TRACK EDITOR

The SEQ Track editor allows you to make alterations to data stored in the current track. When you enter the Track Edit screen, the data contained in the current track is moved to high memory where it is to be manipulated. On exit from the editor the data is returned to low memory. A considerable amount of memory management is performed when this happens and there may be a delay when most of the memory arena is used up.

After recording MIDI data in a track, hold down the SHIFT key and press the E key. The Track edit screen will appear. MIDI events are listed in four columns; Event Time, Event Type, MIDI Channel (channel messages only) and Event Data. At the outside edges of the display area, there are two arrows which indicate the current event. To move the pointer, press the CURSOR DOWN key or SHIFTED CURSOR UP. The screen will scroll up or down as required.

Along the bottom of the screen you will see the functions available through pressing the appropriate function keys; MODIFY, DELETE, INSERT and EXIT. A CTRL-E keystroke will exit without saving any changes made. These operations take effect at the current pointer position. For example, pressing the DELETE option removes the current event from the screen. When INSERT is pressed, a MIDI event list is displayed. Simply press the appropriate key for the type of event that you want to insert. The new event will be inserted into the list at the current position and may now be modified.

The MODIFY option allows you to alter any part of the current event. After pressing F1, a highlighted cursor appears and may be moved between columns by using the CURSOR RIGHT key or SHIFTED CURSOR RIGHT key. The data in the current column may be altered using the + and - keys. You may then leave MODIFY by pressing the RETURN key.

If the event type is a NOTE ON or NOTE OFF, the first data value is shown as a musical note; e.g. C4 (4th octave C). Otherwise, the data shown is a 3-digit decimal number. A NOTE ON event with a velocity value of zero will be shown as a NOTE OFF but should the velocity value be altered to be non-zero then the event will be shown as NOTE ON.

SEQUENCER 64 CONTROL KEY SUMMARY

FUNCTION KEYS - (UNSHIFTED)

F1 - Disk Directory
F3 - Disk Command
F5 - Set PPQN value
F7 - Auto-Correct Current Track

FUNCTION KEYS - (SHIFTED)

F1 - Save Sequence to disk
F3 - Load Sequence from disk
F5 - Save entire song to disk
F7 - Load entire song from disk

FUNCTION KEYS - (COMMODORE KEY)

F1 - Name track
F3 - Name sequence
F5 - Delete track
F7 - Delete sequence

SWITCH CONTROL KEYS

M - Change mode (song/sequence)
C - Clock source (intck/extck)
P - Captive playthrough
X - Record playthrough
I - Record count-in
S - Sync data output
O - Step mode

TAPE TRANSPORT CONTROLS

Cursor Right - Fastforward (1 beat steps)
Cursor Down - Rewind (1 beat steps)
Cursor Left - Fastforward (1 pulse steps)
Cursor Up - Rewind (1 pulse steps)
SPACE BAR - Play
SHIFT/SPACE - Record (current track)
= - Pause
HOME - Rewind to start of sequence

SEQUENCE/TRACK CONTROLS

\+ - Increase tempo
\- - Decrease tempo
1 thru 8 - Mute/Unmute track (also works in play)
SHIFT1thru 8 - Change current track MIDI channel
CTRL 1 thru 8 - Assign sequence to song position
* - Alter sequence repeat value
^ - Alter song pointer position
< - Select previous track as current
> - Select next track as current
SHIFT < - Select previous sequence as current
SHIFT > - Select next sequence as current
CTRL X - Exit Sequencer 64
CTRL C - Clear Sequencer 64 memory
SHIFT E - Enter track editor

TRACK EDITOR CONTROLS

CTRL E - Abort (don't save changes)
Cursor Down - Select next event as current
Cursor Up - Select last event as current
F1 - Modify current event

(IN MODIFY)

\+ - Increment current column value
\- - Decrement current column value
Cursor Right - Select next column as current
Cursor Left - Select previous column as current
Return - Exit modify
F3 - Delete current event
F5 - Insert at current position
F7 - Exit track edi

SECURITY

Put your broken joysticks to good use with this handy utility

Tim Walters

Have you ever broken a joystick trying to score that all important goal using International Football on level nine? If so, this program will put that broken joystick into good use.

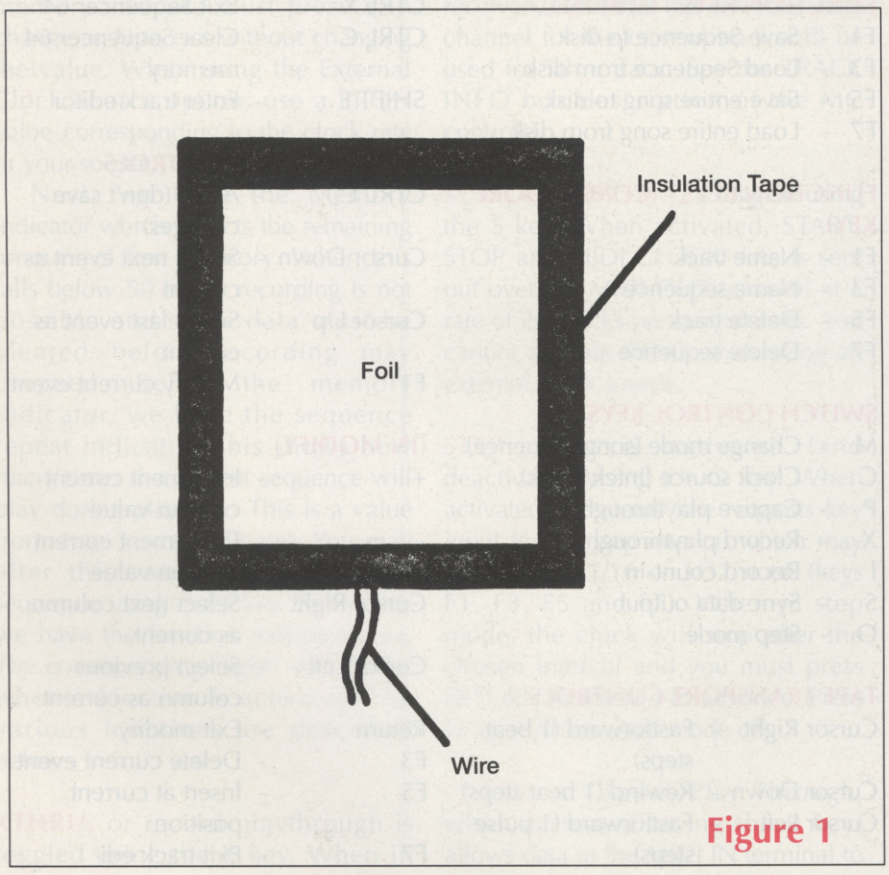


Figure 1

How would you like to make your bedroom, study or even the whole house fully secure using your 64 and a broken joystick? Read on to find out just how to achieve this.

Before getting down to technicalities, a word of warning. NEVER cut the joystick wire while it is still plugged into the computer and NEVER TOUCH THE RED AND BLACK WIRES TOGETHER as this will short circuit the computer and blow a fuse.

To start with, ensure that the joystick is NOT plugged into the computer, cut the wire as close to the joystick as possible. Then, if it is a Quickshot II model, remove the handle and take out the fire button pressure pads by cutting the wires. If it is not a Quickshot and your joystick does not have any pressure pads then use two pieces of tin foil as shown in figure 1. Now trim the seven core wire so that only BLACK and BROWN are exposed and join these wires to pressure pads or foil pieces as shown in figures 2a and 2b.

Once complete load the file SEC V1.00 from the menu. The outer border will flash and a message

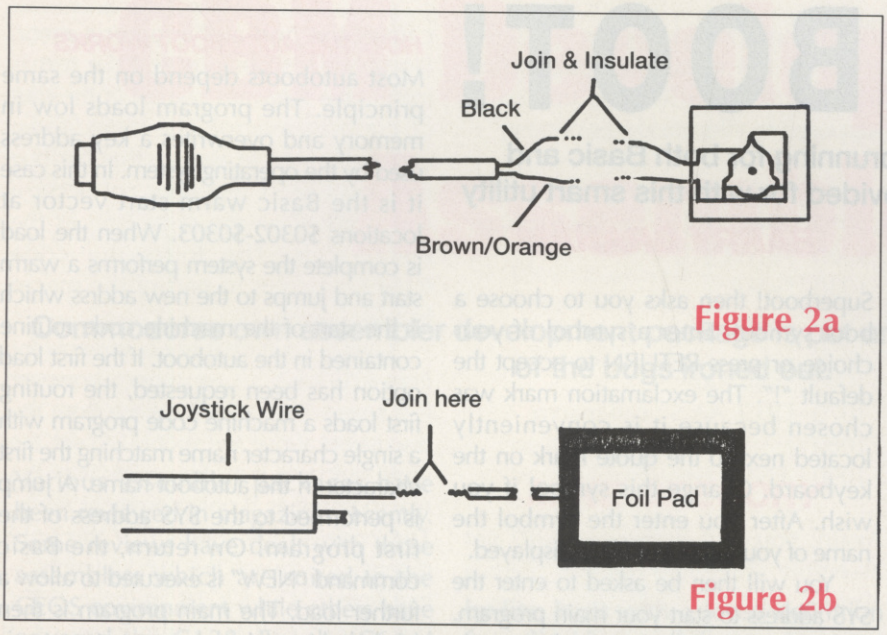


Figure 2a

Figure 2b

'Close door to stop alarm' will be displayed in the centre of the screen. Plug the wire in PORT 1, the screen should continue to flash. If it does not flash then the black and brown wires must be touching. If the screen was flashing then press the pressure pad or touch the foil pieces together, the screen should now stop flashing. If it does not stop, then check the wiring and also check that the plug is in Port 1. If all is correct and the flashing stops then the next stage is installation. A door hinge is an excellent place for both pressure pad and foil, see figure 3. For the wire to reach you might have to extend it. If you are lucky enough to have games good enough to break two joysticks then the second version, SEC V2.00, can also be used.

Remove the pressure pad as before and cut the wire as close to the joystick as possible then trim the seven core wire so that only BLACK and ORANGE are exposed, instead of BLACK and BROWN. Join the bare wires to the pressure pad as shown in figure 2. Load the program SEC V2.00 from the menu. Enter the time when prompted, plug in your first wire {the wire joined to the door} into Port 1 and close your door. If all is correct the screen will display 'door shut. Scanning.....' now plug the new wire into Port 2 and depress the pad, a message 'pressure pad depressed at HHMMSS' should be displayed, if not then recheck your wiring. If all is working then this wire can go under the carpet as shown in figure 4, or some similar place.

Version 3 of Security, SEC V3.00, is identical to version 2 except for one main addition, it stores the information onto disk. This is useful if your mother is prone to turn the computer off as soon as she sees the red light burning. Ensure that you have a blank formatted disk ready and insert it into the drive as soon as the program is loaded. If a pressure pad is depressed or the door is opened or closed, the time and description of the event will be stored on disk, as well as being displayed on the screen.

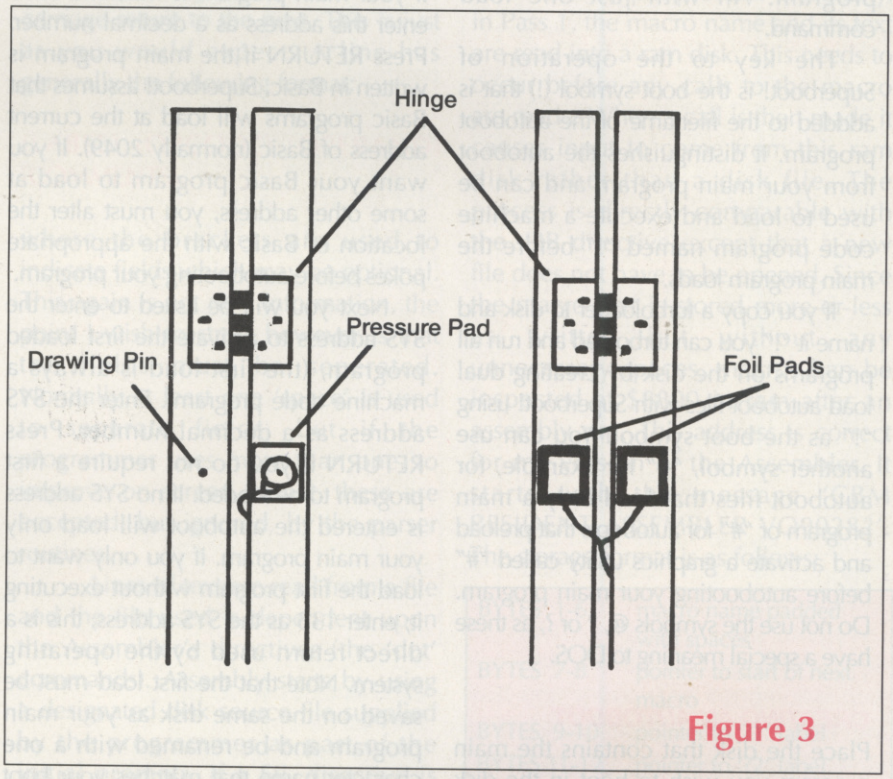


Figure 3

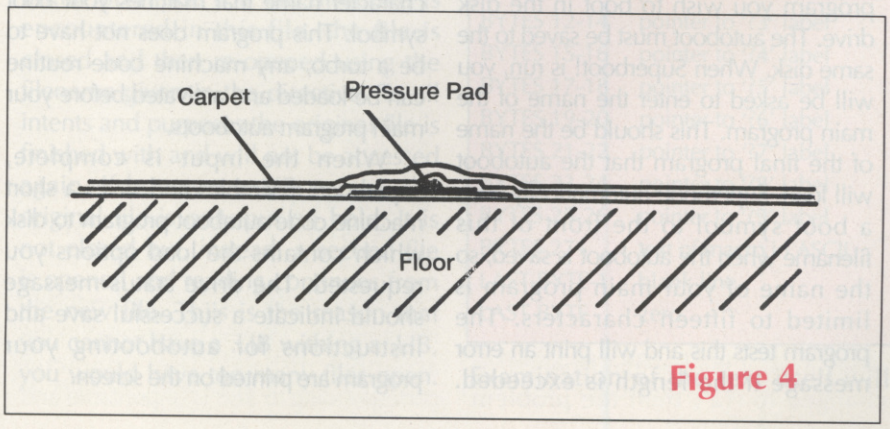


Figure 4

SUPERBOOT!

Single key loading with autorunning for both Basic and Machine Code programs is provided for with this smart utility

BARRY GRAHAM

Superboot! creates a short machine code autoboot program that will load and execute a Basic or Machine Code program automatically from a single load command, just like a commercial program. BUT THAT'S NOT ALL. The autoboot file can load and execute any machine code routine, before loading and running your main program. If you have a disk turboloader that loads into high memory, your boot program will load and activate the turbo before speed loading and running your main program. All with just one load command.

The key to the operation of Superboot! is the boot symbol (!) that is addded to the filename of the autoboot program. It distinguishes the autoboot from your main program and can be used to load and execute a machine code program named "!" before the main program loads.

If you copy a turboloader to disk and name it "!" you can turboload and run all programs on the disk by creating dual load autoboot files with Superboot! using "!" as the boot symbol. You can use another symbol, "+" for example, for autoboot files that load only a main program or "#" for autoboots that preload and activate a graphics utility called "#" before autobooting your main program. Do not use the symbols @, * or ?, as these have a special meaning to DOS.

CREATING AN AUTOBOOT

Place the disk that contains the main program you wish to boot in the disk drive. The autoboot must be saved to the same disk. When Superboot! is run, you will be asked to enter the name of the main program. This should be the name of the final program that the autoboot will load. Superboot! automatically adds a boot symbol to the front of this filename when the autoboot is saved, so the name of your main program is limited to fifteen characters. The program tests this and will print an error message if the length is exceeded.

Superboot! then asks you to choose a boot symbol. Enter a symbol of your choice or press RETURN to accept the default "!". The exclamation mark was chosen because it is conveniently located next to the quote mark on the keyboard. Change this symbol if you wish. After you enter the symbol the name of your autoboot file is displayed.

You will then be asked to enter the SYS address to start your main program. If your main program is machine code enter this address as a decimal number. Press RETURN if the main program is written in Basic. Superboot! assumes that Basic programs will load at the current address of Basic (normally 2049). If you want your Basic program to load at some other address, you must alter the location of Basic with the appropriate pokes before autobooting your program.

Next you will be asked to enter the SYS address to activate the first loaded program, (the first load is always a machine code program). Enter the SYS address as a decimal number. Press RETURN if you do not require a first program to be loaded. If no SYS address is entered the autoboot will load only your main program. If you only want to load the first program without executing it; enter 138 as the SYS address, this is a direct return used by the operating system. Note that the first load must be saved on the same disk as your main program and be renamed with a one character name that matches your boot symbol. This program does not have to be a turbo, any machine code routine can be loaded and activated before your main program autoboots.

When the input is complete, Superboot! assembles and saves a short machine code autoboot program to disk which contains the load options you requested. The drive status message should indicate a successful save and instructions for autobooting your program are printed on the screen.

HOW THE AUTOBOOT WORKS

Most autoboots depend on the same principle. The program loads low in memory and overwrites a key address used by the operating system. In this case it is the Basic warm start vector at locations $0302-$0303. When the load is complete the system performs a warm start and jumps to the new addrss which is the start of the machine code routine contained in the autoboot. If the first load option has been requested, the routing first loads a machine code program with a single character name matching the first character in the autoboot name. A jump is performed to the SYS address of the first program. On return, the Basic command "NEW" is executed to allow a further load. The main program is then loaded, the original warm start vector is reinstated and the end address of the loaded program is stored. If the main program is Basic, the operating system is pointed to the start of the program. The command "CLR" is performed and the program lines are rechained in case the program has been relocated. The boot program then jumps to Basic and the program runs. If the main program is machine code, a jump to the SYS address is performed after the load end address is stored.

RESTRICTIONS

There are a few precautions that should be observed:

The main program to be loaded can be either Basic or machine code, but the filename must not be more than fifteen characters in length.

Your first load program is optional but must be written in machine code if you use on. Enter a SYS address of 138 for the program to load only without executing. The main program should not load over your first load unless this is intentional. NEVER load a main program over a turboloader. Programs that load over locations $02A7-$0303 cannot be loaded by the autoboot. (They autoboot anyway). Relocate Basic manually before autobooting a Basic program that does not load at the normal start of Basic ($0801). Remember to copy your first loaded program to the same disk as your autoboot and main program, then rename it with the first character of your autoboot.

FEATURE

CBM Development Assembler

Commodores own assembler development package is put under the microscope and some of the bugs ironed out.

MIKE GREGORY

Various assembler packages have been reviewed in magazines recently. Some reviews have dealt with those assemblers which are suited to the GEOS environment while others have covered the C128. For the C64, I beleive the best value for money is the Commodre Development Package. It contains a full featured assembler which is not appreciated and consequently not used to its fullest extent. Unfortunately part of the problem is due to the usual standard of Commodore documentation which is poor, and the rest is caused by a few minor bugs! (Note: These are the views of the author of this article and not necessarily those of the magazine..Ed!!)

I have heard a number of complaints from people about not being to use the macro facility and this set me thinking. I have been using macros on this assembler for about 3 years without any problems, so why didn't they work for others? The complaints, not being specific, made it difficult to comment. Recently however I have been putting together some of my programs and, in the process, reviewing and reworking the macros. In effect I was putting together a macro library when assembly errors started to appear. I now understand what the problems are! I can however offer solutions which may be of interest to other users of this excellent development package.

In trying to explain the bugs, a basic understanding of how the Macro Assembler works is required. Essentially, assembly occurs on a line-by-line basis with a loose definition for a 'line' being all the text from one carriage return to the next. This is just as you would expect. A line has generally the following format;

(LABEL) (OPCODE) (OPERAND) (COMMENTS)

where the brackets are used to indicate fields which may be optional. This again is not new information, the point I wish to stress however is that the fields must be seperated. Normally at least one 'space' is used to separate fields, but if the programmer uses more than one, to improve on screen format, these are accepted, but ignored, by the parser routines.

Lines of text are read from a file and the file used is dependent upon the Assembler's directives (the 'dot' commands). Assembly starts by using a designated disk source file supplied by the programmer as part of the initial prompts. If a FIL directive is encountered in this file, the file is closed and then re-opened using the filename given in the directive. To all intents and purposes the original file is finished with and will not be accessed again. If however a LIB occurs, the original file is only put on hold. It is not closed but, instead, a second file is opened and reading continues from the new file. Thjis is the reason that you cannot have a .LIB withing a .LIB, you would have too many files open. Once the .LIB file has been completed to its .END, the file is closed and input is redirected to come from the original file.

The macro facility is designed to fit in with this input process. Essentially, in Pass 1, the macro name and its text are read into a ram disk. This needs to occur before any calls to the macro are made. When a call is then made it causes input to come from this ram disk rather than a disk file. The process is directly comparable with the .LIB directive except that a new file does not have to be opened. Since the macro text is stored more-or-less as written but without any unnecessary spaces, macros ban be inspected at $8800 in ram after an assembly run. This address is correct for my version of the Assembler. It starts with the message 'CBM RESIDENT ASSEMBLER VO90282'. The storage format is as follows:

BYTES 1-6	macro name padded with spaces
BYTES 7-8	pointer to start of next macro
BYTES 9-10	pointer to '?1' label
BYTES 11-12	pointer to '?2' label
BYTES 12-14	pointer to '?3' label
BYTES 15-16	pointer to '?4' label
BYTES 17-18	pointer to '?5' label
BYTES 19-20	pointer to '?6' label
BYTES 21-22	pointer to '?7' label
BYTES 23-24	pointer to '?8' label
BYTES 25-26	pointer to '?9' label
BYTES 27+	text of macro in ASCII
LAST BYTE-1	blank line
LAST BYTE	zero

Examination of the text itself will

show that fields are separated by a single space and that lines end in a carriage return. When a line starts with an opcode, an extra space is added to produce an indent. As is normal, all of the pointers are given in low byte/high byte format. For the parameter values the storage format is a zero-ending name. Where a parameter does not exist, its pointer is $0000. Also since the internal symbols.labels (?1-?9) are filled out, whilst being read in during pass 2, the same macro can produce different expanded code depending upon the supplied parameters! More on this later. One other point worth mentioning is that a macro is recognised by the parser by being a label having an assigned value of $FFFF. This means that you cannot have a label at this address.

Now on to the bugs themselves. Here's a short macro, called PAUSE, that should count to 64K. The code fragment is incomplete and not intended to execute on its own, it serves only to demonstate BUG 1.

```
1000  ;BUG1  DEMO
1010  .MAC   PAUSE
1020         LDY #0
1030         LDX #0
1040  ?1     INX
1050         BNE ?1
1060         INY
1070         BNE ?1
1080  .MND
1090  ;
1100  *=$C000
1110         PAUSE
1120  .END
```

You'll find it hard to believe that this simple piece of code produces so many errors during assembly! Try it. You should get one 'non-alphanumeric' error and four 'undefined symbol' errors. Do the assembly without entering an object filename, (at the first prompt just enter CR), to speed things up. I'll supply a better fix later, but can you note for now that if line 1110 is changed to;

```
1110    PAUSE ;call macro
```

then assembly proceeds without error! Also, if the internal label '?1' is changed to '?2', then the problem disappears. I can use an example given in the menu to demonstrate BUG 2. The following macro is supposed to allocate storage space;

```
1000  ;BUG 2 DEMO
1010  .MAC DECL ; declared storage
1020  ?1 .WOR 0
1030  *=*+?2
1040  .MND
1050  ;
1060  *=$C000
1070  DECL AA,5 ; call macro
1080  ;
1090  .END
```

Again, you will find that it will not assemble. This time we are given a single 'non-alphanumeric' error message. Unfortunately there is now simple fix along the lines given above for this bug. We must do a little patching.

Before fixing these problems it is a worthwhile exercise to examine their causes. A hex memory dump for the macro area after attempting to assemble BUG 1 demo shows the following;

```
8800 50 41 55 53 45 20   ; padded name
8806 48 88               ; start of next
                           macro
8808 0F 9C               ; pointer to ?1
                           label
880A 00 00
880C 00 00
990E 00 00 etc
At address $9C0F we find
9C0F 48 88 00            ; text to be
                           inserted for ?1
```

```
8800 44 45 43 4C 20 20   ; padded name
8806 2B 88               ; next macro
8808 07 9C               ; pointer to ?1
880A 0A 9c               ; pointer to ?2
880C 88 19               ; no other parameters
881A 58 58 3F 31 20 88 57 0D  ; text of line 1
8822 2A 3D 2A 2B 3F 32 0D     ; line 2
8829 0D                  ; blank line
992A 00                  ; end macro

9C07 41 41 00            ; ?1 label
9C0A 35 00               ; ?2 value
```

Observant readers (all) will immediately recognise the 'H' which appeared in the assembly listing. The character corresponding to $88 is unprintable and causes the non-alphanumeric error. However it is obvious that the text is merely the foreward pointer to the start of the next macro! The clues given earlier allow us to deduce that the bug only involves ?1 and does not occur if text is given on the same line as the macro call. The following code fragment relates to part of the macro handler (see fig. 1).

I am not sure that my comments for lines #2C0f-$2C18 are correct, however it is somewhat irrelevent since from the discussion above we should expect the same outcome if no files exists after the macro name as if only comment exists. The code above shows that the only difference between the two situations is that lines $2C0f-$2C18 are executed if the macro call is commented or if parameters are supplied, but not otherwise. The actual bug occurs by having incorrect entry conditions for the subroutine at $2C87. We can correct this by re-ordering the lines;

```
2C0A AD 78 09    LDA $0978
2C0D 8D 80 09    STA $0980
2C10 AD 79 09    LDA $0979
2C13 8D 81 09    STA $0981
2C16 20 D1 15    JSR $15D1
2C19 90 15       BCC $2C30
```

A simpler alternative would appear to be;

```
2C0D 90 24       BCC $2C33
```

which cuts out the subroutine call entirely when there is no field following the macro call. The effect of BUG 2 can be seen by examing the macro storage area at $8800 after assembling the demo. A memory dump is as follows;

As can be seen from the text of the first line the .WOR directive text is missing. A spurious value of $88 together with the W seem to have

```
2BF9 A9 01      LDA #$0       ; set ($10) to point to
2BFB 20 CE 2C   JSR $2CCE     ; parameter pointer
2BF3 A0 11      LDY #$11
2C00 A9 00      LDA #$00      ; set parameter pointers
2C02 91 10      STA ($10),Y   ; to zero
2C04 88         DEY
2C05 10 FB      BPL $2C02
2C07 8D 82 09   STA $0982     ; zero parameter counter
2C0A 20 D1 15   JSR $15D1     ; first char after macro
2C0D 90 21      BCC $2C30     ; branch if no char
2C0F AD 79 09   LDA $0978     ; set up storage pointers
2C12 8D 80 09   STA $0980     ; for internal labels
2C15 AD 79 09   LDA $0979
2C18 8D 81 09   STA $0981
2C1B 20 4F 2C   JSR $2C4F     ; get first char in field
2C1E 90 10      BCC $2C30     ; branch if semi colon
2C20 C9 2C      CMP #$2C
2C22 F0 06      BEQ $2C2A     ; branch if comma
2C24 20 F1 29   JSR $29F1     ; evaluate parameter value
2C27 4C 1B 2C   JMP $2C1B     ; repeat for next parameter
2C2A 20 87 2C   JSR $2C87     ; parameter to zero
2C2D 4C 1B 2C   JMP $2C1B     ; repeat for next parameter
2C30 20 87 2C   JSR $2C87     ; parameter to zero
2C33 60         RTS           ; and end
```

Figure 1

been picked up. The remainder of this particular macro seems to be ok. The values for the parameters are also ok. The following code fragment is the part of the parser routine that reads the macro defination into the ram disk macro storage during pass 1 (Fig. 2).

The start of the bug is fairly obvious. The routine at $2AC3 to store the character ought to be saving the full stop which indicated the assembler directive. The accumulator has however been overwritten by preserving the storage pointers! This part of the problem can be fixed by using the Y index register to preserve the pointers.

```
2AB7 AC 76 09   LDY $0976
2ABA 8C 7C 09   STY $097C
2ABD AC 77 09   LDY $0977
2AC0 8C 7D 09   STY $097D
```

The rest of the problem is at $2ADD where, if the first character after the full stop is not an 'M', since Y is zero the remainder of the line is ignored! This is fixed by redirecting the branch at $2AA8 so as to continue reading the same file;

```
2ADD D0 C9      BNE $2AB1
```

If all of the above corrections are made, both demo macros should assemble without errors and a few useful macros can now be developed. I will now show you how macros are coded and will also demonstrate some other features of the Assembler Package which are either not well documented or are not known.

USING THE CBM MACRO ASSEMBLER

Earlier in this article I showed how the bugs in the CBM Macro Assembler could be fixed. If you have made the necessary corrections to the code we can now proceed to develop and use the macro facility. It is a feature well worth using because of its effectiveness in allowing your coding to be developed as modules.

Listing 1 is a small 'equates' library. I would strongly urge you to build up a similar file of your own. The label names are generally the same as those used in published books such as the Programmers Reference Guide or the Complete Commodore Inner Space Anthology. It should give ready access to common entry points used in the Basic and Kernal ROMs. Notice that, since the file is to be used as a library and will be called by the .LIB directive, it finishes with a .END.

```
1000  ;listing 1
1010  ;equates
1020  ;Mike Gregory July 1989
1030  ;
1040  CRLF=$0D
1050  CLS=$9E
1070  ;
1080  >BASIC ROM ROUTINES
1090  STROUT=$AB1E
1100  LINPRT=$BDCD
1110  ;
1120  >KERNAL ROM ROUTINES
1130  SETLFS=$FFBA
1140  SETNAM=$FFBD
1150  OPEN=$FFC0
1160  CLOSE=$FFC3
1170  CHKIN=$FFC6
1180  CLRCHN=$FFCC
1190  CHRIN=$FFCF
1200  PLOT=$FFF0
1210  ;
1130  .END
```

The routines listed are described in most machine language text books so I will not dwell on them here. Lines 1080 and 1120, which start with a '>' symblol may be of interest. They are a form of hidden comment. The lines will list from the Editor but will not appear in the assembly listing!

Seven macros are supplied in listing 2. Macro 1 and Macro 4, I have indicated how to use conditional assembly feature. The conditional directives are .IFE and .IFN which translate to 'if equal to zero' and 'if not equal to zero' respectively. Both refer to the expression in the field immediately following the directive. Note that the field is separated by at least one space. The conditional text must appear within the '<' and '>' symbols. They are easy to remember if you see them as a pair of stylised braces. The opening symbol is the first field after the expression and on the same line whereas the closing symbol is placed at the end of the text as the first symbol on a new line. The effect of conditional assembly in the first macro is that the code will only be generated if the program counter (*) is at the start of basic ($0801) when the macro is called. That is when *=$0801 is zero. Macro 4 is more complex. Depending on whether or

```
2AA3 A8 FF      LDX #$FF        ; set line pointer
2AA5 8E 6E 08   STX $086E
2AA8 20 3F 2B   JSR $2B3F       ; read first char
2AAB 90 E6      BCC $2A93       ; next line if no char
2AAD C9 2E      CMP #$2E        ; test if full stop
2AAF F0 06      BEQ $2AB7       ; branch if directive
2AB1 20 CA 29   JSR $29CA       ; else store char
2AB4 4C A8 2A   JMP $2AA8       ; and get next
2AB7 AD 76 09   LDA $0977       ; preserce pointers
2ABA 8D 7C 09   STA $097C
2ABD AD 77 09   LDA $0977       ; to macro storage
2AC0 8D 7D 09   STA $097D
2AC3 20 CA 29   JSR $29CA        ; store char
2AC6 A0 00      LDY #$00        ; zero counter
2AC8 20 3F 2B   JSR $2B3F       ; get next char
2ACB 90 C6      BCC $2A93        ; next line if none
2ACD 8D 7E 09   LDA $097E       ; temp save
2AD0 20 CA 29   JSR $29CA       ; store char
2AD3 AD 7E 09   LDA $097E       ; recover it
2AD6 D9 66 20   CMP $2066,Y     ; test for MND
2AD9 F0 22      BEQ $2AFD       ; continue next
2ADB C0 01      CPY #$01        ; not MND
2ADD D0 B4      BNE $2A93       ; next line unless .M
2ADF C9 41      CMP #$41        ; test if A
2AE1 D0 C5      BNE $2AA8       ; next char if not
2AE3 20 3F 2B   JSR $2B3F       ; get next char
2AE6 90 AB      BCC $2A93       ; next line if none
2AE8 C9 43      CMP #$43        ; test if C
2AEA D0 C5      BNE $2AB1       ; branch if not C
2AEC AD 7A 09   LDA $097A
2AEF 8D 76 09   STA $0976
2AF2 AD 7B 09   LDA $097B
2AF5 8D 77 09   STA $0977
2AF8 A9 26      LDA #$26        ; exit with .MAC
2AFA 4C 47 2C   JMP $2C47       ; in .MAC error
2AFD C8         INY             ; test for .MND
2AFE C0 03      CPY #$03
2B00 D0 C6      BNE $2AC8       ; fall thru if .MND
```

Figure 2

not a zero is supplied as parameter 4 (?4), different code is produced! It should be stressed however that in order for this to work, if a label is to be used for the name address, then the label must be defined before the macro is called. Put simply, the assembler needs to know during Pass 1 which code section will be generated otherwise the counter which is used to assign values to labels will get lost. Macro 6 demonstrates how to call a macro from within a macro. It is no different to calling a macro in normal code. One important general point is that the supplied parameters are seperated by commas. Do not use spaces in them to sepearte them. The handler code at $2BF9 takes a space to be the end of the parameter field! This will cause an 'undefined symbol' error.

```
1000  ; LISTING 2
1010  ; MACROS
1020  ; MIKE GREGORY JULY 1989
1030  ;
1040  ; MACRO 1 TO CREATE
      BASIC LINE
1050  ; CALL - BASIC
1060  ;
1070  .MAC BASIC
1080  .IFE *=$0801 <;only
assemble if at $0801
1090  .WORD ?1,10
110   .BYTE
      SYS,'2061',0,10,sys2061
1110  ?1 .WORD 0
1120  >
1130  .MND
1140  ;
1150  ; MACRO 2 TO PRINT
      MESSAGE ON SCREEN
1160  ; CALL - PROMPT ADDRESS
OF TEXT
1170  ;
1180  .MAC PROMPT
1190  LDA #<?1
1200  LDY #>?1
1210  JSR STROUT
1220  .MND
1230  ;
1240  ;MACRO 3 TO GET STRING
FROM INPUT DEV
1250  ;CALL - STRIN STORAGE
      ADDR, ALLOWED LENGTH
1260  ;
1270  .MAC STRIN
1280  LDY #0
1290  ?3 JSR CHRIN ; get char
1300  CMP #CRLF
1310  BEQ ?4
1320  STA ?1+1,Y
1330  INY
1340  CPY #?2
1350  BCC ?3
1360  ?4 STY ?1 ;save length received
1370  .MND
1380  ;
1390  ;MACRO 4 TO OPEN FILE
1400  ;CALL - FOPEN
      NUMBER,DEV,SEC,NAME
      ADDR
1410  ;
1420  .MAC FOPEN
1430  LDA #?1
1440  LDX #?2
1450  LDY #?3
1460  JSR SETLFS
1470  ;
1480  .IFE ?4<;assemble if no name
1490  LDA #0
1500  JSR SETNAM
1510  JSR OPEN
1520  >
1530  ;
1540 . IFN ?4 <;assemble if name
1550  LDA ?4
1560  LDX #<?4+1
1570  LDY #>?4
1580  JSR SETNAM
1590  JSR OPEN
1600  >
1610  ;
1620  .MND
1630  ;
1640  ;MACRO 5 TO CLOSE FILE
1650  ;CALL - FCLOSE NUMBER
1660  ;
1670  .MAC FCLOSE
1680  LDA #?1
1690  JSR CLOSE
1700  .MND
```

```
1710  ;
1720  ;MACRO 6 TO READ DISK
ERROR CHANNEL
1730  ;CALL - FERROR ERROR
      BUFFER
1740  ;
1750  .MAC FERROR
1760  FOPEN 15,8,15,0;open channel
1770  LDX #15
1780  JSR CHKIN;make input file
1790  STRIN ?1,40;read message to
buffer
1800  LDA #CRLF
1810  STA ?1
1820  STA ?1+1,Y
1830  INY
1840  LDA #0
1850  STA ?1+1,Y
1860  FCLOSE15;close file
1870  PROMPT ?1;print message
1880  .MND
1890  ;
1900  ;MACRO 7 TO POSITION
      CURSOR
1910  ;CALL - SURSET XPOS,YPOS
1920  ;
1930  ,MAC CURSET
1940  CLC
1950  LDY #?1
1960  LDX #?2
1970    JSR PLOT
1980    .MND
1990 ;
2000 .END
```

Listing 3 puts it all together in a short program.

```
 1000   ;listing 3
 1010   ;program to demonstrate macros
1020  ;mike gregory july 1989
1030  ;
1040  .OPT NOLIST
1050  .LIB LISTING 1
1060  .LIB LISTING 2
1070  .OPT LIST
1080  ;
1090  *=*0801
1100  BASIC;run start
1110  ;
1120  JMP CODE;climb over storage
1130  ;
1140  ADDR .BYTE 0
1150  .WORD
1160  INBUFF .BYTE 0
1170  *=5+40
1180  ;
1190  CODE PROMPT TEXT;ask for input
1200  STRIN INBUFF,16;get input
1210  CURSET0,12;about midscreen
1220  FOPEN2,8,2,INBUFF;open file
1230  LDX #2
1240  JSR CHKIN;make file input
1250  STRIN ADDR,2;read two bytes
1260  FCLOSE2;close file
1270  FERROR INBUFF;read error
      channel
1280  JSR CLRCHN;restore normal i/o
1290  LDA INBUFF+1
1300  ORA INBUFF+2
1310  CMP #'0'
1320  BNE EXIT;no address if file error
1330  LDA ADDR+2
1340  LDX ADDR+1;convert hex to dec
1350  JSR LINPRT;and print
1360  EXIT RTS
1370  ;
1380  TEXT .BYT CLS,CRLF,'THIS
      ROUTINE'
1390  .BYTE 'WILL DETERMINE THE'
1400  .BYTE 'LOAD',CRLF,'ADDRESS
      OF'
1410  .BYTE 'A PROGRAM',CRLF,
      CRLF
1420  .BYTE 'PLEASE ENTER FILE
      NAME',0
1430  ;
1440  .END
```

The use of the .OPT directive in lines 1040 and 1070 is worth mentioning. You may not always want the full list of equates or all the macros printed in the assembly listing. This is how the listing is switched off. Also the default settings will cause the macros not to be expanded in the assembly listing. Only the macro call will be printed. This is sufficient for macros which have been tried and tested but during development an expansion of the code is often required. This expansion is switched on and off by the GENERATE and NOGENERATE options as is;

```
.OPT GEN
or .OPT NOG
```

Remember that this class of directive can be used in combination;

```
.OPT LIST,GEN
or .OPT NOL,NOG
```

One other assembler bug that come to mind from listing 3 is that you should never add a comment field on the same line as either a .FIL or .LIB directive. The comment will cause a 'file not found' error! I have a fix for this, but the situation is easily avoided. Any avid readers wishing to fix it themselveds will find the relevant code at $11B0. You should consider what happens when a 'space' occurs after the filename. This piece of code also prevents the use of filenames which have spaces in them!

Listing 4 is a short demo for how the Assembler will produce different coding depending upon the form of the supplied parameters. Again the code is not intended to do anything other than demonstrate the point. It should be remembered that the parameter text is supplied verbatim to the parser routines.

```
1000 ;listing 4
1010 ;demo
1020 ;mike gregory july 1989
1030 ;
1040 .MAC MULTI
1050 LDA ?1
1060 .MND
1070 ;
1080 ;NOW TEST IT
1090 .OPT GEN
1100 R6510=$01
1110 *=$2000
1120 ;
1130 MULTI #0;immediate
1140 MULTI 128;zero page
1150 MULTI 256;absolute
1160 MULTI LABEL;absolute
1170 MULTI #>R6510;immediate high
byte
1180 MULTI #<LABEL;immediate low
byte
1190 ;
1200 LABEL .BYTE 0
1210 ;
1230 .END
```

The assembly listing will show the different codes which are generated.

I hope you find this article of some use and value. Its purpost is not to provide a program that finds a file load address but to demonstrate what can be done with the CBM Assembler, and, in particular with macros. It should show that quite complex programs can be developed speedily by using a good, sound macro library. The ability to get an idea up and running quickly often helps to maintain the initial enthusiasm. Once it is running, the code can be optimised for memory space or for execution speed if required.

TECHNO INFO

Our resident agony aunt sifts through more of your technical queries.

Dear CDU,
I am considering buying an Amiga 500 but wish to know if I can use it with my Commodore 1902 colour monitor which I currently use with a C128D. The 1902 has composite video, RGB and RF sockets and so I guess that at least one of these would be compatible with the Amiga. However, I am not sure if the resolution would be as good as the new 1084 monitor used with the Amiga.

R.J.Cranton, Somerset.

Dear Mr.Cranton,
Both the 1084 and 1902 monitors are medium resolution monitors and so I can see no reason for the Amiga not to produce a decent image on your particular monitor. It must allow an analog RGB display and have a separate connection for the audio output, unless of course you plan to link it with a stereo system. If there is any difference in resolution then I can only think that it would be so slight as to be unnoticeable.

Dear CDU,
I write to you concerning the excellent Hi-Res Demo Kit by Neil Higgins (Jul/Aug 1989) and the subsequent plea for help in the February edition of Techno Info. The disk contained a program which creates a file that then loads the demo created, and also loads and executes a BASIC program once the spacebar has been pressed. The utility works well with the original demo contained on the disk which has been compacted - it does not however work with any demo created in its original format. Also could you please tell me how to use character sets created by Font Factory 89 in my own programs once they have been saved to disk. Perhaps it would be worth having a program support section where you give details or hints on how results can be best achieved when using CDU such as how to incorporate the utilities for your own use.

J.Mullen, Coatbridge.

Dear Mr.Mullen,
Thanks for pointing this error out - I admit defeat! I must sincerely apologise to Scott Mathieson who originally requested the program and to all other readers who have tried this out. Hopefully this sort of thing will never happen again. Now that grovelling is over, how to rectify the fault - it is a simple matter of changing one number and is related to the machine code starting address of the demo. The compacted one had a start address of 2080 and normal ones use 2115. Therefore, load the program PROB1 from the disk and then list line number 215. Change the first part so that the data reads: 215 DATA 0,32,213,255, 32,67,8,... It is the 67 that is different. Then save it back to the CDU disk. With regard to Font Factory you should add the following lines to the start of your program: 1 A=A+1: IF A=1 THEN LOAD "filename",8,1 and then to enable the new set include a line like 2 POKE 53272,28 in your program. I think that telling the users how to operate a particular program and utilise it fully is really up to the programmer himself. The whole purpose of the instructions in his contribution text is to not only reveal how to use the program, but also how to incorporate things created by it in your own programs. Therefore, if anyone is considering submitting anything please remember that not everyone knows everything about your program and so it is quite important that you cover every detail that people need to know. As for a program support section, that is part of the service that Techno Info offers. Should anyone, like yourself, have a query about any program that CDU has published then simply drop us a line at the address given at the end. Thanks once again for pointing out my mistake - no supper for me tonight!

Dear CDU,
With my C64, 1541 and MPS801 I use Precision Softwares SuperScript word-processor for correspondence. To improve printing quality and produce Norwegian letters I bought an Olympia ESW1000C daisywheel printer said to be compatible with the C64. I cannot, however, find anyone to tell me how to make the ESW1000C print from the word-processing program. Having discovered your magazine recently I hope that you can advise me.

Knut Sjovorr, Orpington.

Dear Knut,
Unfortunately the only thing that I can suggest is that, with SuperScript and printer manuals in hand, you go very carefully through the defaults file that is supplied with the word-processor and change everything relevant according to what your printer manual states. This file covers all sorts of things and is the only way that I can think of that the program can be configured to your printer. Sorry that I cannot be of any more help.

Dear CDU,
First of all many thanks for a very good disk magazine. The next reason for writing is to ask for some help. I have a C128, 1571 disk drive and the MPS1200P printer and as you

can imagine I am really pleased to see the C128 programs. But I am really fed up this month (June). I have tried every way possible to load them but they will still not load. Is it me, I have tested the computer and it loads other programs without problems, but the program POPP and C128 Converter/Maths Aid will not load. I would be most grateful of any help.

David Taylor, Nuneaton.

Dear David,
I can safely say that it is not you that is at fault, unless you are not using the DLOAD command. You do not mention what the problem with loading is but I suspect that you get a file not found error. This is not uncommon. A lot of 128/1571 setups produce the same result and I have found that it is due to the files being individually write-protected, that is - having a less than sign after the file type in the directory. You must use a disk editor to remove this protection - it is a simple matter of changing a byte. But most directory editors have a facility to unprotect a file. Alternatively use the program I supplied with my Directories Explained article in the February 1990 issue. If that does not work then the next most common cure is to move the file entry to the top of the directory.

Dear CDU,

I have a C128 and 1571 and a 1901 monitor but this months disk (June) seems not to work correctly. I was able to get the games working but the programs that I wanted - no way. Aleatory Music was still announcing that it was loading code after an hour. After attempting to load the Personal Organiser Page Printer in both 1541 and 1571 modes all I got was a file not found error. The C128 Converter/Maths Aid did load successfully but then I was faced with a blank screen for an hour. I would be very grateful if you could point out where the errors lie, whether with my limited experience or with the programming.

W.Nisbet, Plymouth.

Dear Mr.Nisbet,
There were no errors with the software and I cannot see why Aleatory Music and C128 Converter/Maths Aid did not work. Possibly with the latter your monitor was not in eighty column mode although I presume that you tried that. The problem with POPP and the file not found error is as described in the previous letter. If you (and Mr.Taylor) would like to return your disk to me at Techno Info then I shall have a look to see what I can do in the way of rectifying any problems. Please mark your package REQUESTED.

Dear CDU,

In the June 1990 issue of CDU you presented two programs on the disk which I found especially useful - Sprite Generator 2 and Sprite Basic. Although I have been able to use both programs separately, when using Sprite Basic I have not managed to load a sprite definition produced by using Sprite Generator 2. I would be most grateful if you could please tell me what is wrong and supply a routine which would solve my problem. Thanking you in anticipation of your assistance.

Alexander Sendler, Bedford.

Dear Alexander,
It is possible to load and operate Sprite Basic and then to load and use a sprite created with Sprite Generator 2, so I can only assume that you are doing something wrong. Therefore I shall just go over the procedure. Load Sprite Generator 2 and create your shapes. Then save them to a disk, remembering first to reset the drive if you loaded the program from the menu. Then switch everything off and load Sprite Basic. Again, if loaded from the menu you should reset the drive. When you have initialised Sprite Basic load the sprite data remembering the ,8,1 suffix. Then use the relevant commands - SPRXY, SPRAT, SPRSLOT and SPRITE (in that order) - to display the sprite. Make sure you follow very carefully the instructions in the magazine. Your sprite should appear. Perhaps the fault is in using the wrong pointer areas for the sprites in Sprite Generator 2. Sprite Basic requires that a sprite has a pointer (slot) value that is over 32 but also preferably over 128. That means that if, when you save the data from the generator program, you are giving values of less than $2000 or more than $4000 then you are using the wrong areas. Also ensure that the pointer value on the editting screen is 128 or more ($80 in hex). Other than that I can provide no more help. I hope you will be able to get it to work now.

Dear CDU,

With reference to the CDU April 1990 Techno Info, you published a letter from a person named Stuart Smith of Manchester who asked whether it is possible to have an eighty column display on a C64. Actually it IS possible. In the Your Commodore magazine of January 1988 there is a listing of a program to have this facility. The program uses the graphic bitmap screen and one can use the normal BASIC facilities. But there are some restrictions. Only one colour can be used at a time and you have less memory. The screen memory map for instance is twice as large as usual. When I program (in BASIC) I have a little problem with REM statements. It is that I cannot type shifted letters in REM statements. For example, typing 10 REM (shift) C (shift) B (shift) M only results in 10 REM LEN PEEK FOR when the program is listed. Why can I not use shifted letters in REM statements and why are they replaced by BASIC keywords? How can this problem be rectified? Thirdly, I am going to buy a printer but I am not sure which one to buy. I am interested in the Commodore version of the Star LC-10. Is it fully compatible with the C64 and its software, or is it better to buy an MPS1200 or Citizen 120D printer? Also, is it worth buying the colour version and if so is it compatible

LETTERS

with PrintPic that you published in the February issue? I will use it mainly for word-processing but also for program listings and dumping graphic screens. Finally, with regard to PrintPic, and although I do not own a printer, I think that if people load a Koala File picture and then select the Trilogic option for printing, everything will still work. If this does work then owners of the excellent Advanced OCP Art Studio drawing package can also use PrintPic by converting their files using the program TECHNO INFO that was featured on the April 1990 disk. That program converts from OCP to Koala File format. Please let me know if the above really does work and with what type of printers it works. Also could you kindly answer the other questions that I have asked in this rather long letter.
Ludwig Flask, Malta.

Dear Ludwig,
Thankyou very much for your letter all the way from Malta. On your first point I would actually contest you and say that I DID tell Stuart that it was possible - using the bitmap screen as you said. But the programming techniques are far too complex for me to explain it here because for a start all outputs to the screen of any kind have to be rerouted to a routine to calculate the positions and definitions of the characters, and screen editting also becomes very difficult to tackle. If you would kindly forward me a photocopy of the program listing or a disk containing the program then I will forward it on to Stuart. The reason for the keywords appearing is because all BASIC programs are tokenised/encoded using shifted (and some other) characters for the keywords. Therefore when you list the program with these characters in the REM statements they are decoded to the keywords that they represent. To overcome the problem you must put the computer in quote mode. That is the reason for having to enclose any DATA that contains non-alphanumeric characters within quotation marks. Simply by applying the same theory to REM statements you will be able to have the shifted characters. Place one (and only one) quotation mark - shifted two - immediately after the REM command. On your next point about printers I am not going to elaborate too much. It is impossible to say whether or not the Star LC-10, or any other printer for that matter, will be compatible with every piece of software that has been or will be produced. Most things, though, work fine. Whether to choose a Star over an MPS or a Citizen must be decided by you. It is best to get a demonstration of each and make up your mind from there. The Star LC-10 (colour version) is compatible with PrintPic. Your final point is slightly more difficult to answer specifically because I am still waiting for my colour printer to arrive and so I cannot test the program (incidentally, it will NOT work with standard black only ribbon printers). In theory it works because when PrintPic is operated the picture appears with all colours correct (except background - change with F1). I can therefore see no reason why it would not work. Could somebody please try this out (using a Koala File or OCP converted picture with PrintPic) for us and tell us what happens. Be careful with the prefix to Koala File pictures - to obtain the reversed spade symbol use the Commodore key and one. I hope I have been of assistance.

Dear CDU,
I have been trying for a number of weeks to sell my MPS801 printer that is in immaculate condition but everyone around where I live either already has or does not want a printer. So it may be slow and a little noisy in comparison to todays printers but at only forty pounds (which includes postage) you would think people would be clammering for it. It is fitted with a special chip to provide true descenders and three other fonts although the switch for this needs slight attention - a bit of soldering that I could do myself if someone asked nicely. I really cannot understand how anyone without a printer, so long as you do not want it for really professional things, could afford not to buy it. It will even support GEOS and dumps graphic screens with ease. Please, Techno Info, could you help me.
Jason Finch, somewhere in England.

Dear Jason,
Well I certainly cannot understand why no-one wants it. Who really cares if it is unidirectional because if something does its job I say grab it before somebody else does. I, myself, find it very difficult to resist the temptation. If anyone does want to buy this superb printer then please get in touch with us at the address below. You never know, Jason may even accept a tiny bit less depending upon how quick you ask for it. I wish you all the best with selling it Jason.

That about wraps it up for this issue because Tip of the Month is on hold until the September issue. I wanted to provide you with a little program but the big Mr E (alias Paul Eves) refused to grant me any disk space because I was a little late and there was that big hoo-har with CDU and a certain publisher so I did not know whether I was coming or going! And none of you wonderful people have sent me any tips so I could hardly include one of those. How many times do I have to ask for a Tip of the Month from one of you before someone obliges??? If you do want to share your knowledge with us, or if you have any programming problem or query, then please do not hesitate to drop us a line at the NEW Techno Info address: CDU Techno Info, 11 Cook Close, Brownsover, Rugby, Warwickshire, CV21 1NG. Please note that this is only the address for Techno Info. All other correspondence and contributions should be sent to the other address for the magazine which will probably be printed elsewhere in this issue. So until next month - get thinking, get writing, and have fun!

SPECIAL OFFER

SOFTWARE OFFER!!

For all those readers that have missed out on previous issues of CDU, we are providing special compilation disks of a number of programs that have appeared on the disks in the past. The disks each cover a specific topic, ie Sound/Music programming, Machine code programming, Basic programming etc etc. The first two of these new compilation disks are available now. Disk number 1 is a Machine Code special and Disk number 2 is devoted to General Utilities.

To obtain the disks, simply fill out the coupon below and send with your remittance to the address as stated.

This disk is made up of the following programs:

LINK and CRUNCH. Once a machine code program has been written it often occupies several different areas of memory and doesn't necessarily use the space as efficiently as it could. The LINKER and CRUNCHER have been devised to help you correct this and save the need for Basic loaders.

PSYMON. Psymon is a machine code monitor program for use with the C64. There are two versions, one located at 36884 and the other at 49152. This is because you may want to use the area of memory where the monitor is actually located. There are a total of 20 commands available to the user. This is a good monitor for those that are not too experienced in machine code.

LOCATION FINDER. It gets pretty boring looking through object code to find out just which page zero locations it's messing up. Take the legwork out of it with this simple program. Location Finder simply tells you which memory locations a piece of object code is using, including those all important zero page locations. The information it provides could be vital if you want to incorporate a piece of object code into your own programs.

ZMON. We tend to get over obsessed with 6510 programming, forgetting that lurking inside our C128's there is a perfectly good Z80 microprocessor. Zmon makes this processor available to the built-in Machine Language Monitor. No longer is it necessary to import a machine-specific operating system like CP/M to try out the Z80. Just prefix your MONITOR command with 'Z' and ZMON will automatically invoke the Z80 to carry it out.

DISK TURBO. Disk Turbo, once installed, should speed up your disk saves and loads by a factor of around ten times. The only limitation to the program is that it allows for a maximum program of 189 block programs to be Turboloaded, however, this should not present too many problems.

6510+ ASSEMBLER. Use this assembler once and you may never need another aid to writing machine code programs. This assembler is a valuable aid both for writing professional machine code programs and for learning about programming. It is a three pass assembler which allows the use of labels and contains extra commands that speed the production of code by permitting merging routines from tape or disk.

MICKMON. Get to grips with your programming techniques with this powerful Machine Language Monitor. Mickmon is designed for use on a C64 with a 1541 or compatible disk drive. In addition to monitoring the computers own internal state, MICKMON has commands that access the disk drives memory, and can even address the surface of the disk itself. It also has powerful extras not found on other monitor programs.

6510+ UNASSEMBLER. To compliment the 6510+ assembler program, we have provided the unassembler which is specific for this particular assembler. Once used, you will discover just exactly how valuable an unassembler can be as part of your arsenal of programming utilities. An unassembler makes the inspection of machine code programs a pleasure instead of a burden.

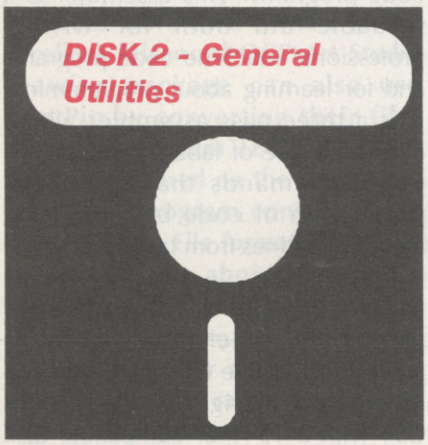
BAR PROMPTS. Professional looking menu driven programs and applications can be achieved by this simple machine code program. All you are ever likely to need to produce these impressive results are contained in this one short program.

SPEEDY UNASSEMBLER. Like the 6510+ unassembler above, this program is specific to the Speedy Assembler that has been produced by Your Commodore for the past 4 years. Finding your way round long

SPECIAL OFFER

forgotten code is now that much easier. A program no serious machine code programmer should be without.

That concludes the list of programs that appear on the first, Machine Code Special, compilation disk.



This disk is made up of the following programs:

DIRECTORY DESIGNER. Sort out your disks with this versatile and powerful disk utility. Typical of Commodore to make such a wonderful computer as the C64 then hide all it's power so that using it is, to say the least, cumbersome. This program allows the user access to the world of the Disk Drive DOS without the need to cram megabytes of knowledge into their head.

DISK LIBRARIAN. This program is best described as a superb disk-filing system, as opossed to program. In essence the program allows the user to order their files on their disks into two large databases. These databases are the Chronological file and Categories files. Keeping tabs on all your programs has never been this easy.

DISK TOOLBOX. This program may well be the ultimate in disk utilities. Just about everything you need is here to give you total disk-based happiness. The program incorporates such things as A Monitor, Extended Basic Commands, Disk Drive Editor, File Copier and extensive disk based utilities to make life easier for the programmer/program developer. Not to be missed.

DATA MAKER. Data maker is a useful machine code utility for converting memory locations to Data statements. This has a varied range of applications. For example, machine code programmers may want to convert their coding to a Basic loader for those who do not have an assembler. Also it could be used to convert sprite data, and add this as DATA statements at the end of the resident Basic program.

DEVAID. Devaid, which is short for Development Aid, adds no less than 41 new commands to the resident C64 Basic. It gives the programmer a very powerful army of new commands to aid the development of programs written in Basic. Some of the commands can be used both in Direct mode as well as Program mode, provided the main program is in memory.

BASE-ED. Base-Ed is a random access database allowing a maximum of 500 records per disk which may be entered then subsequently viewed, rectified, deleted and interrogated. Each record can have a maximum of 39 fields but the record length must not exceed 255 characters.

SCRAPBOOK. This program is a type of database and as its name suggests, it is used rather like the scrapbook you probably had as a child. This may seem fairly unuseful, but for many applications it has advantages over the traditional database organisation. Information is stored not as records with set fields, but rather as pages.

PROGRAM COMPARE. Any Basic program developer knows the headaches that can be caused whenever you modify or alter a large Basic program. The inevitable GOTO or GOSUB that gets missed out. The line you delete by mistake, then forget what it was originally etc etc. All these headaches are a thing of the past. With this program, you can compare different versions of your program, either on screen or via your printer, and instantly see where any corrections, modifications, deletions etc etc occur.

SPREADSHEET 64. A really superb Spreadsheet program for the C64 written in WEOS Basic (Window Environment Operating System). This spreadsheet program will aid any company or household to follow the flow of money. This helps in forecasting figures and in budgeting. The program is easy to use and is very user friendly.

TEXTED. A compact but powerful program which may meet all your wordprocessing needs. Texted is a wordprocessor which provides most of the features found on commercial programs, coupled with easy icon selected commands and advanced printer and disk interaction.

That concludes the list of programs on the second, General Utilities, special compilation disk.

Look out for two more compilation disks, Graphics programming and Sound/Music programming, coming your way shortly.

Please send me ..... Copies Disk No. 1 @ £5.95 each
..... Copies Disk No. 2 @ £5.95 each
I enclose a cheque/postal order for .....
(Made payable to **Alphavite Publications Ltd.**)
Name .....
Address .....
..... Postcode .....
Send to: **Alphavite Publications Ltd.**, CDU Software Offer, 20 Potters Lane, Kiln Farm, Milton Keynes, MK11 3HF.

Commodore DISK USER

Lineage: 53p per word. (+VAT)
Semi display: £11.50 plus VAT per single column centimetre minimum 2cm.
Ring for information on series bookings/discounts.
All advertisements in this section must be prepaid.
Advertisements are accepted subject to the terms and conditions printed on the advertisement rate card (available on request).
Make cheques payable to Alphavite Publications Ltd.
Send your requirements to:
CLASSIFIED DEPARTMENT
ALPHAVITE PUBLICATIONS LTD., 20 POTTERS LANE, KILN FARM, MILTON KEYNES, MK11 3HF.

(0908) 569819

SOFTWARE

SILVER WING SOFTWARE
'THE DEDICATED C64 P.D. LIBRARY
We have the best & latest quality demos, utilities & games available.
ONLY £2.00 on Tape or Disk
For the best benefits & power from your 64, you can not do without Public Domain Software...
For our lists send an SAE to:
SILVER WING SOFTWARE
185 CALLOWBROOK LANE, RUBERY, BIRMINGHAM B45 9TG.

REPAIRS

M.C.E. SERVICES
33, Albert Street, Mansfield, Notts. NG18 1EA Tel: 0623 653512
FIXED PRICE COMMODORE REPAIRS

C64	£24.95
C16	£19.95
PLUS4	£24.95
AMIGA A500	£39.95
1541	£29.95
1571	£31.95
C2N etc	£12.95

All prices inc. VAT, Labour, Parts etc.
NO MORE TO PAY.
Please ring for details.

DISKS

WESTONING LTD

3½ DISKS
DS/DD 46p
DS/HD £1.15
PRICE EACH INC VAT AND LABELS
LOCKABLE BOXES
CAP 40 - £4 CAP 80 - £5.75

5¼ DISKS
DS/DD 23p
DS/HD 57p
PRICE EACH INC VAT, SLEEVES AND LABELS
LOCKABLE BOXES
CAP 50 - £5.25 CAP 100 - £6.50

AMIGA
BATMAN PACK £355
FLIGHT FANTASY PK £355
CLASS OF 90's £510
512K RAM £56
WITH CLOCK £62

NEC EXTERNAL
3½ DRIVE . . . £60
1084D MON £239

PRICES INCLUDE VAT NO QUIBBLE GUARANTEE
POST ADD £3 NEXT DAY ADD £7

12 SANDERSON ROAD, WESTONING, BEDFORD MK45 5JY
Tele. (0836) 775060 or (0525) 718668 Fax (0525) 715789

FULL TERMS AND CONDITIONS ARE AVAILABLE ON REQUEST CALL
0908 569819

SERVICES

COMMODORE SUPPLIES

C64 Power Unit	£21.99
C128 Power Unit	£39.99
C2N/C3N Datasette	£24.99
100 x 3½" Discs	£29.99
C64 Magnum Lightgun	£34.99
C64/128 Serial Cable	£5.99

Prices include VAT & P&P Chq/PO to:
OMNIDALE SUPPLIES (CD), 23 Curzon Street, Derby, DE1 2E6
Tel 0332 291219

CALL 0908 569819 TO ADVERTISE

Commodore DISK USER

Lineage: **53p** per word. (+VAT)
Semi display: **£11.50** (+ VAT) per single column centimetre. Min size 2cm.
Series discounts available on request. All classified advertisements must be pre-paid.
There are no reimbursements for cancellations.
SEND TO CDU CLASSIFIED DEPARTMENT, ALPHAVITE PUBLICATIONS, 20 POTTERS LANE, KILN FARM, MILTON KEYNES, MK11 3HF.

I enclose my cheque/postal order for £ ..... for ..... insertions, made payable to Alphavite Publications.
(Delete as necessary)
PLEASE DEBIT MY ACCESS/VISA CARD NO:

EXP. DATE. .....

£ ..... FOR ..... INSERTIONS

Name .....
Address .....
Postcode .....
Daytime Tel No ..... Signature ..... Date .....

❑ FOR SALE ❑ SOFTWARE ❑ SPECIAL OFFERS ❑ REPAIRS ❑ HARDWARE ❑ DISKS

...it's dynamite!

POWER CARTRIDGE

FOR YOUR COMMODORE 64/128

"...unbelievable value for money" ZZAPP! Dec 89

- POWER TOOLKIT
- POWER MONITOR
- TAPE & DISK TURBO
- PRINTERTOOL
- POWER RESET
- TOTAL BACKUP

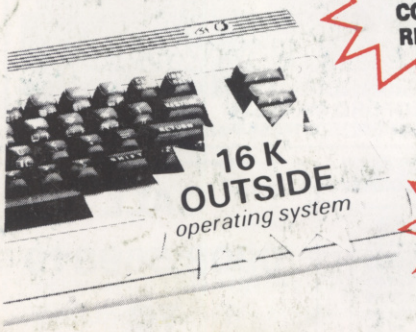
..."Money well spent" YC/CDU Jan 90

42 Pg Manual - "Damned Good Handbook" CCI Jan 90

SO MUCH FOR SO LITTLE

AVAILABLE FROM ALL GOOD COMPUTER RETAILERS

TRIED AND TESTED - OVER 100,000 SOLD IN EUROPE



..."highly recommended for C64 users" CCI Jan 90

YOU WILL WONDER HOW YOU EVER MANAGED WITHOUT IT

POWER TOOLKIT

A powerful BASIC-Toolkit (Additional helpful commands) that considerably simplifies programming and debugging.

AUTO	HARDCAT	RENUMBER
AUDIO	HARDCOPY	REPEAT
COLOR	HEX$	SAFE
DEEK	INFO	TRACE
DELETE	KEY	UNNEW
DOKE	PAUSE	QUIT
DUMP	PLIST	MONITOR
FIND	ILOAD	BLOAD

RENUMBER : *Also modifies all the GOTO's GOSUB's etc. Allows part of a program to be renumbered or displaced.*
PSET : *Set up of printer type.*
HARDCAT : *Prints out Directory.*

The toolkit commands can be used in your programs.

DISK TOOL

Using POWER CARTRIDGE you can load up to 6 times faster from disk.
The Disk commands can be used in your own programs.

BLOAD	DVERIFY	DIR
DSAVE	MERGE	DEVICE
DISK		

MERGE : *Two BASIC programs can be merged into one.*
DISK : *With DISK you can send commands directly to your disk.*

TAPE TOOL

Using POWER CARTRIDGE you can work up to 10 times faster with your data recorder. The Tape commands can be used in your own programs.

LOAD	SAVE	VERIFY
MERGE	AUDIO	

POWERMON

A powerful machine language monitor that is readily available and leaves all of your Commodore memory available for programming.
Also works in BASIC-ROM, KERNAL and I/O areas.

A ASSEMBLE	I INTERPRET	S SAVE
C COMPARE	J JUMP	T TRANSFER
D DIS-ASSEMBLE	L LOAD	V VERIFY
	M MEMORY	W WALK
F FILL	P PRINT	X EXIT
G GO	R REGISTER	$ DIRECTORY DOS Commands
H HUNT		

PRINTERTOOL

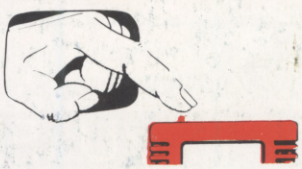
The POWER CARTRIDGE contains a very effective Printer-Interface, that self detects if a printer is connected to the Serial Bus or User-Port.
It will print all Commodore characters on Epson and compatible printers.
The printer-interface has a variety of set-up possibilities. It can produce HARDCOPY of screens not only on Serial printers (MPS801, 802, 803 etc) but also on Centronic printers (EPSON, STAR, CITIZEN, PANASONIC, etc).
The HARDCOPY function automatically distingishes between HIRES and LORES. Multi-colour graphics are converted into shades of grey. The PSET functions allow you to decide on Large/Small and Normal/Inverse printing.
The printer PSET functions are:

PSET 0 - *Self detection Serial/Centronics.*
PSET 1 - *EPSON mode only.*
PSET 2 - *SMITH-CORONA mode only.*
PSET 3 - *Turns the printing 90 degrees!!*
PSET 4 - *HARDCOPY setting for MPS802/1526.*

PSET B - *Bit-image mode.*
PSET C - *Setting Lower/Upper case and sending Control Codes.*
PSET T - *All characters are printed in an unmodified state.*
PSET U - *Runs a Serial printer and leaves the User-port available.*
PSET Sx - *Sets the Secondary address for HARDCOPY with Serial Bus.*
PSET L1 - *Adds a line-feed, CHR$ (10), after every line.*
PSET L0 - *Switches PSET L1 off.*

Bitcon Devices Ltd does not authorise or purport to authorise the making by any means or for any purpose whatsoever of copies or adaptions of copyright works or other protected material, and users of the Power Cartridge must obtain the necessary prior consent for the making of such copies or adaptions from all copyright and other right owners concerned. See UK Copyright, Designs & Patents Act 1988.

POWER RESET



On the back of the POWER CARTRIDGE there is a Reset Button. Pressing this button makes a SPECIAL MENU appear on the screen.
This function will work with many programmes.

CONTINUE - *Allows you to return to your program.*
BASIC - *Return to BASIC.*
RESET - *Normal RESET.*
TOTAL BACKUP DISK - *Saves the contents of the memory onto a Disk. The program can be reloaded later with BLOAD followed by CONTINUE.*
RESET ALL - *RESET of any program.*
TOTAL BACKUP TAPE - *As BACKUP DISK but to TAPE.*
HARDCOPY - *At any moment, prints out a Hardcopy of the screen. Using CONTINUE afterwards you can return to the program.*
MONITOR - *Takes you into the Machine language Monitor.*

BDL
Bitcon Devices Ltd

88 BEWICK ROAD
GATESHEAD
TYNE AND WEAR
NE8 1RS
ENGLAND

Tel: 091 490 1975 and 490 1919 Fax 091 490 1918
To order: Access/Visa welcome - Cheques or P/O payable to BDL
Price: £16.99 incl. VAT.
UK orders add £1.20 post/pack total - £18.19 incl. VAT.
Europe orders add £2.50. Overseas add £3.50
Scandinavian Mail Order and Trade enquiries to: Bhiab Elektronik, Box 216, Norrtalje 76123, SWEDEN. Tel: + 46 176 18425 Fax: 176 18401
TRADE AND EXPORT ENQUIRIES WELCOME